



The University of
Nottingham

UNITED KINGDOM • CHINA • MALAYSIA

Hybrid Methods for Modelling Advanced Electromagnetic Systems using Unstructured Meshes

Daniel Simmons

Supervisors:

Kristof Cools, Phillip Sewell, Steve Greedy

Thesis submitted to The University of Nottingham
for the degree of Doctor of Philosophy, May 2016

Abstract

The aim of this project is the conception, implementation, and application of a simulation tool for the accurate modeling of electromagnetic fields within inhomogeneous materials with complex shapes and the propagation of the resulting fields in the surrounding environment. There are many methods that can be used to model the scattering of an electromagnetic field, however one of the most promising for hybridisation is the Boundary Element Method (BEM), which is a surface technique, and the Unstructured Transmission Line Modeling (UTLM) method, which is a volume technique. The former allows accurate description of the scatterer's boundary and the field's radiation characteristics, but cannot model scattering by materials characterized by a non-uniform refraction index. The latter, on the contrary, can model a very broad range of materials, but is less accurate, since it has to rely on approximate absorbing boundary conditions. A method resulting in the hybridisation of BEM and UTLM can be used to construct a tool that takes into account both the interaction with non-uniform tissue and propagation in its environment. The project aims to describe in detail the implementation of the novel method, and deploy it in a heterogeneous distributed computing environment.

Acknowledgements

Whilst pursuing this PhD, I have received support from a number of individuals. I owe gratitude to all those people who have made this thesis possible, who have made my postgraduate experience forever treasured.

My deepest gratitude goes out to Dr Kristof Cools who has been a mentor, a colleague, and a friend. I have been extremely fortunate to have had such an approachable supervisor who, despite my initial shortcomings, had the capability to educate me to such a high degree. Kristof's guidance has been pivotal in transforming an idea to a completed study, and has made the whole journey thoroughly rewarding.

I am deeply grateful for Professor Phillip Sewell's insightful comments and constructive criticisms at different stages of my research which helped generate fresh ideas. His thirst for debates on technical aspects of the project was inspiring (and so too were his discussions on bureaucracy, and motor-bikes)!

A very special thanks also goes out to Dr Steve Greedy, Dr Ana Vukovic, and the rest of the academics at the George Green Institute for Electromagnetics Research for all their valued support throughout the years.

The encouragement of my co-workers during my PhD, even through the almighty stress of buying my first house, has been amazing. Thank you Ahmed Elkalsh (shukran!), Sendy Phang (Makasih!), Vicky Meng (xiè xie!), Latifah (Terima kasih!), Afonso Bernardino (obrigado!), Edward Zhang (xiè xie!), Najla Najeeb

(nandri!), Soumitro Kumar (dhanyavaad!), Gihan (istuti!), and Hayan Nasser (cheers!); I am extremely lucky to have met you all, and honoured that you made the perilous journey from co-worker to friend. I must also acknowledge my girlfriend, Becky, without whose care and understanding, especially during my mother's passing, I would not have finished this thesis.

Most importantly, none of this would have been possible without the love and patience of my close family; my grandfather who has been the foundation throughout my life, my grandmother who would be proud regardless, and my mother, whom this thesis is dedicated to.

To you mum.

May you rest in peace.

Publications related to this thesis

D. Simmons, K. Cools, and P. Sewell, “Coupling of unstructured TLM and BEM for accurate 2D electromagnetic simulation,” in *2015 Int. Conf. Electromagn. Adv. Appl.*, (Torino), pp. 1076-1079, IEEE, September 2015.

D. Simmons, K. Cools, and P. Sewell, “Modelling antennas in outer space using the boundary element unstructured transmission-line (BEUT) method,” in *36th ESA Antenna Work. Antennas RF Syst. Sp. Sci.*, (Noordwijk), 2015.

D. Simmons, K. Cools, and P. Sewell, “A Hybrid Boundary Element Unstructured Transmission-line (BEUT) Method for Accurate 2D Electromagnetic Simulation,” *J. Comput. Phys.*, no. [Submitted for publication], 2016.

Contents

Abstract	iii
Acknowledgements	v
List of Publications	vii
List of Symbols	xiii
List of Abbreviations	xv
1 Introduction	1
1.1 Background	1
1.2 Modeling Techniques	2
1.3 Motivation for hybridising the Boundary Element Method and the Unstructured Transmission-Line Modeling method	4
1.4 Implementation proposal	8
1.5 Synthesis of thesis	9
References	10
2 Electromagnetic Theory	15
2.1 Maxwell's Equations	15
2.2 Representation Formulas	17
2.2.1 The Green's Function	20
2.2.2 1D Representation Formulas	23
2.2.3 2D Representation Formulas	25
References	29
3 The Boundary Element Method	31
3.1 Jump conditions	32

3.2	Equivalence Principle	33
3.3	Scattering by a Perfect Electric Conductor	34
3.4	Scattering by a Penetrable Object	35
3.5	Scattering by 2 Spatially Distinct Penetrable Objects	36
3.6	Reducing the Singularity	38
3.7	Implementation	39
3.7.1	Testing functions	40
3.7.2	Piecewise Polynomials	42
3.7.3	Spatial basis functions	44
3.7.4	Discretised Equations	45
3.7.5	Gaussian coefficient table	46
3.7.6	Basis function index tables	49
3.7.7	Temporal basis functions	51
3.7.8	Temporal convolutions	52
3.8	Validation	57
3.8.1	Scattering by a Perfect Electric Conductor	57
3.8.2	Scattering by a Penetrable Cylinder	59
3.8.3	Scattering by 2 Penetrable Cylinders	61
	References	64
4	The Unstructured Transmission-Line Modelling Method	65
4.1	1D TLM Theory	66
4.2	2D UTLM Theory	67
4.3	Implementation	77
4.3.1	Scatter process	79
4.3.2	Connect process	81
4.3.3	Connection at the boundaries	82
4.4	Validation	84
	References	87
5	Review of Previous Hybridisation Attempts	89
5.1	The TLM-IE method	89
5.2	The Adapted Radiating Boundaries (ARB) method	92
5.3	Modelling thin wire structures	95
5.4	IRIS (Interference and Radiation of Internal Surfaces) method	96
5.5	Total-Field/Scattered-Field (TF/SF) Technique	97
5.6	Hybridisation of 2D FD-TLM with 2D-EFIE	98
5.7	Comparisons and Conclusions	99

References	102
6 The Boundary Element Unstructured Transmission-line Method	105
6.1 Theory	106
6.1.1 1D Comparison	108
6.1.2 The Boundary Element Unstructured Transmission-line Representation Formulas	109
6.2 Implementation	110
6.2.1 Basis functions	110
6.2.2 Stability	112
6.2.3 Code	112
6.2.4 Parallelisation	113
6.3 Alternative Methods of Hybridisation	114
References	115
7 Results using BEUT	117
7.1 Validation	117
7.1.1 Free space cylinder excited with plane wave	117
7.1.2 Free space cylinder excited with a point source at differ- ent locations	118
7.1.3 Free space cylinder excited with a point source at differ- ent frequencies	118
7.2 Accuracy: Dielectric cylinder excited with a plane wave	121
7.3 Speed: Two Spatially Distinct Dielectric Cylinders	124
7.4 Canonical test cases	124
7.4.1 Two Spatially Distinct Lüneburg Lens Antennas	126
7.4.2 Dipole Antenna and Radome Interaction	127
References	132
8 Conclusions	133
8.1 Overview of the work presented	133
8.2 Future work	135
References	136
A BEUT Matlab Implementation Manual	137
A.1 BEUT	140
A.2 Meshing	141
A.3 Excitation	146
A.4 UTLM	149

A.5	BEM	152
B	BEM C++ Implementation Manual	161
B.1	Installation	162
B.1.1	Linux	162
B.1.2	Windows	163
B.2	Usage	164
B.2.1	Examples	165
B.2.2	Initial test	165
C	BEUT Tutorial	167

List of Symbols

Symbol	Description
t	Time
i	Imaginary unit
ω	Angular frequency
φ	arbitrary vector
e	Electric field intensity
h	Magnetic field intensity
d	Electric flux density
b	Magnetic flux density
j	Electric current density
m	Magnetic current density
ρ_e	Electric charge
ρ_m	Magnetic charge
ε	Electric permittivity
ε_0	Free-space permittivity
ε_r	Relative permittivity
μ	Magnetic permeability
μ_0	Free-space permeability
μ_r	Relative permeability
k	Wavenumber
G	Green's function
Ω_∞	Exterior region
u	Symbol denoting E or H
u^i	Incident field
\mathbf{r}'	Source location
\mathbf{r}	Observation location
δ	Dirac delta function
Γ	Surface of volume
Ω	Volume/domain
$\hat{\mathbf{n}}$	Normal unit vector
Γ_0	Object boundary
Γ_∞	Exterior region boundary

Symbol	Description
u^s	Scattered field
R	Distance
ϵ	Infinitesimal distance
c	Speed of propagation
H	Heaviside function
$\hat{\mathbf{t}}$	Tangential unit vector
D, D', S	2D BEM operators
N, N_h, N_s	
S	Spatial basis function
T	Temporal basis function
N_T	Number of timesteps
N_F	Number of basis functions
D, D', S	Discretised BEM operators
N, N_h, N_s	
G	Discretised Gram matrix
d	Polynomial coefficient
p	Polynomial degree
P	Max. polynomial degree
α	Basis function index
L	Inductance
C	Capacitance
R	Resistance
V	Voltage
I	Current
χ	Number of functions in a piecewise polynomial
Q	Number of Gaussian quadrature points
q	Current Gaussian quadrature sample
w	Quadrature weight
β	Test function index
Φ	Oversampling factor
H_n	Hankel function
J_n	Bessel function of order n
σ	Conductivity
ϕ_α	Angle between link lines opposite port α
Δ_α	Link length at port α
X	Expansion coefficient
Δt	Timestep
Γ	Reflection coefficient
η_0	Characteristic impedance of free space
c_0	Speed of propagation in free space

List of Abbreviations

EM	Electromagnetic
EMI	Electromagnetic interference
EMC	Electromagnetic compatibility
TD	Time domain
FD	Frequency domain
DE	Differential equation
IE	Integral equation
FEM	Finite Element Method
FDTD	Finite-Difference Time-Domain
BEM	Boundary Element Method
TLM	Transmission-Line Modeling
ABC	Absorbing Boundary Conditions
MoM	Method of Moments
BEUT	Boundary Element Unstructured Transmission-line
PMCHWT	Poggio-Miller-Chan-Harrington-Wu-Tsai
1D, 2D, 3D	One, two, three dimension
LHS	Left Hand Side
RHS	Right Hand Side
TM	Transverse magnetic
TE	Transverse electric
EFIE	Electric Field Integral Equation
MFIE	Magnetic Field Integral Equation
CFIE	Combined Field Integral Equation
PEC	Perfect Electric Conductor
PML	Perfectly Matched Layer
ARB	Adapted Radiating Boundaries
TD-AIM	Time Domain Adaptive Integral Method
PWTD	Plane Wave Time-Domain

Introduction

The first chapter of the thesis will describe the background of the research, including current electromagnetic modelling techniques, how these techniques are implemented in practice, and the motivation for carrying out the proposed project. The thesis as a whole will then be introduced.



1.1 Background

The dawn of the twenty-first century brought the rise of electromagnetic (EM) wireless technology. Real world wireless appliances have complex and irregular features, so their electromagnetic fields cannot be analytically calculated. However, computational and numerical techniques have the ability to model a vast range of applications. The results of these computational simulations allow optimum design of radar systems, medical imaging, cell-phone antennas etc. and also check for electromagnetic interference (EMI) and compatibility

(EMC). Time domain numerical solvers in the field of electromagnetics have evolved rapidly, on the one hand due to the rapid increase in computational power over the last few decades and on the other hand due to a number of breakthroughs in the efficient solution of linear systems that lie at the heart of the algorithms. Realistic scenarios which involve fields travelling through inhomogeneous and complex targets positioned in large free space environments are difficult to simulate accurately and efficiently with traditional schemes.

1.2 Modeling Techniques

There are several numerical methods which model scattering of an electromagnetic field. Equation 1.1 shows the generic model of the scattering problem. [1.1]

$$L\{f(x)\} = g(x) \tag{1.1}$$

where $g(x)$ is the stimulus function and $f(x)$ is the response function to be found.

Variable x can be in time-domain (TD) or frequency-domain (FD). TD methods are best for transients, wide-band applications and non-linear problems, whereas FD methods are best for steady-state, narrow-band applications [1.2].

Operator L can be a differential equation (DE) where operations on every point in the problem space is required, or an integral equation (IE) where operations on the surfaces (or volumes) of the problem space are enforced [1.2]. The former has the ability to detail complex geometrical features but is computationally expensive, whereas surface integral methods are computationally cheaper but cannot deal with intricate geometries or inhomogeneous materials.

Popular methods used in computing the scattering of electromagnetic fields include the Finite Element Method (FEM), the Finite-Difference Time-Domain

(FDTD) method, the Boundary Element Method (BEM), and the Transmission-Line Modeling (TLM) method.

Also known as the Method of Moments (MoM), BEM is a surface technique which allows accurate description of the scatterers boundary and the fields radiation characteristics, but cannot model scattering by non-homogeneous media. The BEM method solves Maxwell's equations in differential form using jump conditions and the wave equation which are then formulated as integral equations [1.3]. The technique uses less computational resources for problems with a small surface to volume ratio (where the volume is homogeneous, ideally free space). The method is commonly used as a standard for checking other methods and is efficient at solving problems with wire-like structures.

There are time domain volume integral equation techniques that are formulated in the same manner as BEM, but can model inhomogeneous objects. However, they are generally computationally expensive as the whole volume is discretised (rather than just the surface) [1.4], but there have been recent advancements that have improved the efficiency [1.5] and further enhancements that enable investigation of time-varying media also look promising [1.6,1.7].

FEM is a well-established numerical modeling technique for finding approximate solutions to partial differential equations through numerical discretisation. It is traditionally formulated in the FD, although TD formulations have also been developed [1.8]. The technique can be applied to complex geometries with varying material coefficients and boundary conditions [1.9], but can be inefficient when dealing with highly conducting radiators when compared to BEM [1.8].

FDTD is a well-established finite difference method and, as the name suggests, belongs to the class of time-domain differential methods. Maxwell's equations in partial differential form are discretised and the resulting finite-difference equations are solved in a leapfrog manner; the electric field vector components are found then the magnetic field vector components are found at the next instant in time. The process is repeated until the desired electromagnetic field behaviour is determined. In this way, the technique is intuitive and extremely easy to implement whilst providing animated displays of electromag-

netic movement through the model. This has led to the availability of many commercial packages that use this technique. The disadvantages include the fact that the entire domain has to be meshed and the electromagnetic response of the medium must be modeled explicitly, resulting in a large computational domain and poor quality of approximations between grid points. The technique in it's structured form suffers from staircasing, and dispersive materials require considerable effort to implement [1.8].

The TLM method is based on the analogy between an electromagnetic field and a mesh of transmission lines. TLM is a powerful time-domain, volume method which and has the ability to model a very broad range of three-dimensional structures and materials, including complex, non-homogeneous materials with time-varying properties. However, just like FDTD, the technique suffers from staircasing, has to rely on approximate absorbing boundary conditions (ABCs) and is computationally expensive as the entire problem space is discretised.

1.3 Motivation for hybridising the Boundary Element Method and the Unstructured Transmission-Line Modeling method

Techniques that are the amalgamation of two numerical methods to combine the best features from both are called hybrid methods. These methods can be very efficient, but are difficult to design and implement and so there are not many available commercial packages that take advantage of hybrid methods.

Many advancements are constantly being applied to integral equation (surface) techniques, and differential equation (volume) techniques. By creating a hybrid solver in an elegant manner, we can enjoy the advantages of each technique, without modifying the underlying derived methods. We can use a volume technique to model the fields inside the scatterers, and a surface technique to model the external field interactions and radiation conditions.

The Time Domain (TD) Boundary Element Method (BEM) can be solved by a marching-on-in-time technique which gives an accurate description of outwardly radiating fields because it uses the explicit expression for the Green's function to represent these radiating fields. The method decreases the dimensionality of the problem by one dimension which, when combined with modern matrix-vector product acceleration techniques, leads to a very efficient method (for homogeneous materials). For details of the state-of-the-art discretisation and implementation of marching-on-in-time space-time Galerkin methods, the reader is referred to, for example, [1.10–1.14].

The Unstructured Transmission-Line Modeling (UTLM) method is a time domain volume technique which is unconditionally stable for all time (subject to a maximum timestep constraint) and can model complex, non-linear materials with complex geometries. It is based on core engineering principles, from which a physical insight into the propagation of waves is retained. Curved surfaces can be represented in the mesh with much higher accuracy than is possible with the Cartesian meshing required of structured TLM, thus avoiding staircasing errors. Furthermore, there is a large code base of legacy implementations that have proven their worth in academia and industry. Unfortunately just like the structured TLM, the UTLM method requires the use of Approximate Boundary Conditions (ABCs) to model the radiating behaviour of the fields at the boundary of the simulation domain. Moreover, for these ABCs to be accurate, the simulation domain needs to be extended beyond the domain occupied by the device under study, leading to an increase in the size of the problem and thus an increase in solution time [1.2,1.15]. Finally, the ability to model plane wave excitations and compute radar cross sections, though possible, is not immediately available using TLM [1.16].

UTLM can be compared to other time domain volume techniques such as the Finite-Difference Time-Domain (FDTD) and the Finite Element Method (FEM) which are both well-established numerical modelling methods. These methods also have the ability to model inhomogeneous and complex media [1.17–1.19], but only FEM is naturally constructed for unstructured grids (though FDTD has been extended to non-orthogonal and unstructured meshes previously [1.20,1.21]). FDTD has the advantage of being exceptionally simple to implement and FDTD meshes can be terminated with very good absorbing boundary conditions. However, the exact location of boundaries can be prob-

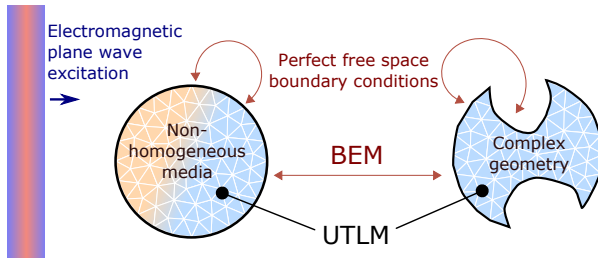


Figure 1.1: Arbitrary objects modelled using UTLM, separated by free space modelled by BEM.

lematic due to the offset nature of the electric and magnetic field grids, and the appealing simplicity is lost when attempting to apply FDTD to unstructured meshes. FEM can naturally handle complex geometries and dispersive materials, and has the ability to model multi-physics applications, however it is more difficult to implement compared to FDTD, and its meshes can become very complex [1.8]. Unlike UTLM and FDTD, FEM is an implicit time-marching scheme i.e. the solution of a linear system via a matrix inversion is computed at each time step, which if directly solved is computationally expensive. Iterative solvers can be used, which have roughly linear memory and compute requirements, but rely on the appropriate use of dedicated preconditioners. Alternatively, the sparse matrix seen in FEM can be approximated to a diagonal matrix via “mass lumping”, but this technique can give an unstable algorithm which depends heavily on the problem [1.22].

As with UTLM, FDTD and FEM do not include radiation conditions for open regions. This is overcome by hybridizing with an efficient integral equation technique.

Fig. 1.1 shows a typical example of a device comprising complex materials and complex geometries inside spatially distinct and well separated regions. Scattering by and transmission through such a device is most efficiently modelled by a method hybridising the UTLM and the BEM methods. The advantages of this hybridised scheme are:

- Modeling of complex, non-linear media [1.23] with geometrically complex features

- Perfect radiating boundary conditions
- Straightforward excitation by plane waves
- Free space region does not need to be meshed, enabling a significant decrease in the degrees of freedom and more efficient computation of open boundary problems

The novel hybrid method described in this paper is called the Boundary Element Unstructured Transmission-line (BEUT) method. It is conceptually very simple and can be easily applied to existing solvers of the two underlying methods. In fact its derivation is directly linked to the construction of the Poggio-Miller-Chan-Harrington-Wu-Tsai (PMCHWT) integral equation for the modelling of transmission problems through piecewise homogeneous devices [1.24]. The key ingredient is the construction of a representation formula valid on the (inner) boundary of the TLM governed regions.

Previous hybridisations between TLM and BEM have been attempted [1.25–1.31]. However these solvers either contain complicated connection processes that require a large number of discrete Green’s functions, contain discretisation errors on subdomain boundaries (therefore requiring padding between the object and the TLM/BEM interface), or require the use of the inaccurate TLM ABCs. They do not take advantage of unstructured meshes or take into account recent advancements that make BEM and TLM more robust, stable and accurate. These techniques are reviewed in chapter 5.

There are other hybrid techniques that couple with BEM such as FDTD-BEM [1.32,1.33] and FEM-BEM [1.34,1.35]. Unlike FEM and FDTD, UTLM uses a physical discretization which can translate to a network of lumped elements. By using circuit theory, the method demonstrates the educational appeal of TLM, as its workings can be understood at an early stage using concepts that are available in a classic electrical engineering programme. Also, the transmission line description of the low frequency response of the domain automatically guarantees stability during runtime, i.e. the output energy equals the input energy, which is especially useful for large simulations.

A UTLM-BEM hybrid has advantages over a FDTD-BEM technique, since

most FDTD implementations rely upon a structured mesh, limiting the freedom in meshing and the accuracy of boundary representation. The advantages of UTLM-BEM over FEM-BEM are more subtle. On the one hand, many existing codes are based on structured and unstructured TLM. The ability to directly couple these codes to a BEM solver should definitely be seen as an advantage. On the other hand, UTLM lends itself to the relatively easy inclusion of more exotic media such as meta-materials, cells containing wires, and active media [1.36–1.39]. Finally, UTLM has more straightforward stability properties than FEM. In fact, for passive lossless media, the Euclidian norm of the solution vector at different time steps is exactly conserved. In FEM stability is regulated by bounds that contain hard to estimate constants and that depend on the spatial meshing and temporal oversampling. Given the sensitivity of the stability of TD-BEM solvers, coupling to the trivially stable UTLM is considered to be a more conservative choice.

It must be noted that there are methods to obtain good absorbing boundary conditions through the use of a Perfectly matched Layer (PML) in TLM [1.40–1.42], however these methods are still inferior to PMLs previously implemented in FDTD [1.43,1.44]. The use of BEM to truncate the UTLM mesh gives more accurate boundary conditions that can be applied directly to the surface of the scatterers and also allows spatially distinct scatterers to interact without modelling the space between. However, it requires computation of a global interaction matrix for all surface elements at each timestep. This means that the hybrid method is more efficient only for transient scattering problems involving large free space regions, and where accuracy is vital.

1.4 Implementation proposal

The project aims to deploy a novel hybridisation of the two dimensional (2D) BEM and 2D UTLM methods in a distributed computing environment. To achieve this, an array of computational techniques will be required, including the use of coding languages C++ and MATLAB, and also the use of the Armadillo matrix library. For execution across many processors, the Open MultiProcessing (OpenMP) interface is used, and for portability of the code, makefiles are created with the help of CMake’s build system. Other spe-

cialist software has also been used such as GIT which assists in organizing and versioning source code besides collaboration management via GitHub, which hosts all the published code related to this project online at <https://github.com/dan-phd>.

For now, we use a straightforward implementation of BEM and UTLM which have computational complexities of $\mathcal{O}(N_S^2 N_t^2)$ and $\mathcal{O}(N_V N_t)$ respectively, where N_V and N_S denote the number of spatial field sampling points within the volume and on the surface of the scatterer respectively, and N_t is the number of timesteps. However, the BEM implementation could be further accelerated using techniques such as the Time Domain Adaptive Integral Method (TD-AIM) [1.45,1.46] which uses a spatial and temporal Fast Fourier Transform (FFT) for computing convolutions, and the Plane Wave Time-Domain (PWTD) algorithm [1.47,1.48] which aggregates far-fields from source sub-scatterers into plane waves which are then superimposed onto the observer.

1.5 Synthesis of thesis

This thesis will begin by introducing Maxwell's equation and deriving the one dimensional (1D) and 2D representation formulas in chapter 2. From these equations, the frequency domain (FD) and thus time domain (TD) BEM Poggio-Miller [1.49] style equations will be derived along with implementation guidelines in 3. The 1D TLM and 2D UTLM formulations will be derived in chapter 4. A review of previous attempts to couple BEM and TLM will be discussed in chapter 5. Comparisons will be made between 1D TD BEM and 1D TLM, which will consequently be used when hybridising the 2D methods, which will be described in chapter 6. Results obtained using the novel method for validation and demonstration will be shown in chapter 7. Conclusions and advice on future work will be given in chapter 8.

References

- [1.1] R. F. Harrington, *Field Computation by Moment Methods*, ser. IEEE PRESS Series on Electromagnetic Wave Theory. IEEE PRESS, 1993.
- [1.2] C. Christopoulos, *The Transmission-Line Modeling Method TLM*, ser. The IEEE Series on Electromagnetic Wave Theory. Wiley, 1995.
- [1.3] K. Cools, “Spectral Properties of Boundary Integral Equations: Analysis and Regularization,” Ph.D. dissertation, Ghent University, 2008.
- [1.4] N. T. Gres, A. A. Ergin, E. Michielssen, and B. Shanker, “Volume-integral-equation-based analysis of transient electromagnetic scattering from three-dimensional inhomogeneous dielectric objects,” *Radio Sci.*, vol. 36, no. 3, pp. 379–386, may 2001.
- [1.5] B. Shanker, K. Aygün, and E. Michielssen, “Fast analysis of transient scattering from lossy inhomogeneous dielectric bodies,” *Radio Sci.*, vol. 39, no. 2, pp. n/a–n/a, apr 2004.
- [1.6] A. Nerukh, P. Sewell, and T. Benson, “Volterra Integral Equations for Nonstationary Electromagnetic Processes in Time-Varying Dielectric Waveguides,” *J. Light. Technol.*, vol. 22, no. 5, pp. 1408–1419, may 2004.
- [1.7] N. Sakhnenko, A. Nerukh, T. Benson, and P. Sewell, “Investigation of 2-D electromagnetic transients in a circular cylinder with time discontinuity in permittivity via the resolvent method,” *Opt. Quantum Electron.*, vol. 39, no. 10-11, pp. 825–836, aug 2007.
- [1.8] D. B. Davidson, *Computational Electromagnetics for RF and Microwave Engineering*, 2nd ed. Cambridge University Press, 2010.
- [1.9] S. Zaglmayr, “High Order Finite Element Methods for Electromagnetic Field Computation,” Ph.D. dissertation, Johanne Kepler University of Linz, 2006.
- [1.10] Y. Beghein, K. Cools, H. Bagci, and D. De Zutter, “A Space-Time Mixed Galerkin Marching-on-in-Time Scheme for the Time-Domain Combined Field Integral Equation,” *IEEE Trans. Antennas Propag.*, vol. 61, no. 3, pp. 1228–1238, mar 2013.
- [1.11] D. Weile and G. Pisharody, “A novel scheme for the solution of the time-domain integral equations of electromagnetics,” *Antennas ...*, vol. 52, no. 1, pp. 283–295, 2004.
- [1.12] D. S. Weile, “Accelerating convolution quadrature,” in *2015 Int. Conf. Electromagn. Adv. Appl.* Turin: IEEE, sep 2015, pp. 341–344.
- [1.13] B. Shanker, A. Arif Ergin, K. Aygün, and E. Michielssen, “Analysis of transient electromagnetic scattering phenomena using a two-level plane wave time-domain algorithm,” *IEEE Trans. Antennas Propag.*, vol. 48, no. 4, pp. 510–523, 2000.

- [1.14] M. Lu, E. Michielssen, B. Shanker, and K. Yegin, “Fast Time Domain Integral Equation Solvers for Analyzing Two-Dimensional Scattering Phenomena; Part I: Temporal Acceleration,” *Electromagnetics*, vol. 24, no. 6, pp. 425–449, jan 2004.
- [1.15] P. Sewell, J. Wykes, T. Benson, C. Christopoulos, D. Thomas, and A. Vukovic, “Transmission-line modeling using unstructured triangular meshes,” *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 5, pp. 1490–1497, 2004.
- [1.16] F. J. German, G. K. Gothard, L. S. Riggs, and A. N. D. P. M. Goggans, “THE CALCULATION OF RADAR CROSS-SECTION (RCS) USING THE TLM METHOD,” vol. 2, no. August 1989, pp. 267–278, 1990.
- [1.17] F. L. Teixeira, “Time-domain finite-difference and finite-element methods for Maxwell equations in complex media,” *IEEE Trans. Antennas Propag.*, vol. 56, no. 8 I, pp. 2150–2166, 2008.
- [1.18] J. Paul, C. Christopoulos, and D. Thomas, “Generalized material models in TLM .I. Materials with frequency-dependent properties,” *IEEE Trans. Antennas Propag.*, vol. 47, no. 10, pp. 1528–1534, 1999.
- [1.19] —, “Generalized material models in TLM .II. Materials with anisotropic properties,” *IEEE Trans. Antennas Propag.*, vol. 47, no. 10, pp. 1535–1542, 1999.
- [1.20] A. Taflov and S. C. Hagness, *Computational Electrodynamics The Finite Difference Time Domain Method*, 3rd ed. Artech House, 2005.
- [1.21] S. D. Gedney, S. Member, and J. A. Roden, “Numerical Stability of Nonorthogonal FDTD Methods,” *IEEE Trans. Antennas Propag.*, vol. 48, no. 2, pp. 231–239, 2000.
- [1.22] Lee Jin-Fa, Lee Robert, and A. Cangellaris, “Time-domain finite-element methods,” *IEEE Trans. Antennas Propag.*, vol. 45, no. 3, pp. 430–442, 1997.
- [1.23] J. Paul, C. Christopoulos, and D. W. P. Thomas, “Generalized material models in TLM .III. Materials with nonlinear properties,” *IEEE Trans. Antennas Propag.*, vol. 50, no. 7, pp. 997–1004, 2002.
- [1.24] Y. Beghein, K. Cools, F. P. Andriulli, D. De Zutter, and E. Michielssen, “A calderon multiplicative preconditioner for the PMCHWT equation for scattering by chiral objects,” *IEEE Trans. Antennas Propag.*, vol. 60, pp. 4239–4248, 2012.
- [1.25] L. Pierantoni, S. Lindenmeier, and P. Russer, “A Combination of Integral Equation Method and FD/TLM Method for Efficient Solution of EMC Problems,” *27th Eur. Microw. Conf. 1997*, vol. 2, pp. 937–942, 1997.
- [1.26] S. Lindenmeier, L. Pierantoni, and P. Russer, “Adapted radiating boundaries (ARB) for efficient time domain simulation of electromagnetic interferences,” *1998 IEEE MTT-S Int. Microw. Symp. Dig. (Cat. No.98CH36192)*, vol. 2, pp. 465–468, 1998.

- [1.27] —, “Hybrid Space Discretizing - Integral Equation Methods for Numerical Modeling of Transient Interference,” *IEEE Trans. Electromagn. Compat.*, vol. 41, no. 4, pp. 425–430, nov 1999.
- [1.28] —, “Numerical modelling of transient radiated interferences in time domain by the hybrid ARB method,” *Int. J. Numer. Model. Electron. Networks, Devices Fields*, vol. 12, no. 4, pp. 295–309, jul 1999.
- [1.29] S. Lindenmeier, C. Christopoulos, and P. Russer, “Methods for the modeling of thin wire structures with the TLM method,” *2000 IEEE MTT-S Int. Microw. Symp. Dig. (Cat. No.00CH37017)*, vol. 1, pp. 387–390, 2000.
- [1.30] M. Zedler and G. V. Eleftheriades, “Anisotropic transmission-line metamaterials for 2-d transformation optics applications,” *Proc. IEEE*, vol. 99, no. 10, pp. 1634–1645, oct 2011.
- [1.31] M. Naser-Moghadasi, M. Bahadorzadeh, and R. Sadeghzadeh, “Implementation of a Novel TLM-MOM Hybrid Method for the Analysis of Interference in Antennas,” *2008 3rd Int. Conf. Inf. Commun. Technol. From Theory to Appl.*, pp. 1–4, apr 2008.
- [1.32] M. Lu, M. Lv, A. A. Ergin, B. Shanker, and E. Michielssen, “Multilevel plane wave time domain-based global boundary kernels for two-dimensional finite difference time domain simulations,” *Radio Sci.*, vol. 39, no. 4, 2004.
- [1.33] B. Shanker, M. Lu, A. A. Ergin, and E. Michielssen, “Plane-wave time-domain accelerated radiation boundary kernels for FDTD analysis of 3-D electromagnetic phenomena,” *IEEE Trans. Antennas Propag.*, vol. 53, no. 11, pp. 3704–3716, 2005.
- [1.34] D. Jiao, A. Ergin, B. Shanker, E. Michielssen, and Jian-Ming Jin, “A fast higher-order time-domain finite element-boundary integral method for 3-D electromagnetic scattering analysis,” *IEEE Trans. Antennas Propag.*, vol. 50, no. 9, pp. 1192–1202, sep 2002.
- [1.35] A. E. Ylmaz, Z. Lou, E. Michielssen, and J.-m. Jin, “A Single-Boundary Implicit and FFT-Accelerated Time-Domain Finite Element-Boundary Integral Solver,” *IEEE Trans. Antennas Propag.*, vol. 55, no. 5, pp. 1382–1397, may 2007.
- [1.36] P. Sewell, Y. K. Choong, and C. Christopoulos, “An accurate thin-wire model for 3-D TLM simulations,” *IEEE Trans. Electromagn. Compat.*, vol. 45, no. 2, pp. 207–217, 2003.
- [1.37] S. Phang, A. Vukovic, H. Susanto, T. M. Benson, and P. Sewell, “Ultrafast optical switching using paritytime symmetric Bragg gratings,” *J. Opt. Soc. Am. B*, vol. 30, no. 11, p. 2984, 2013.
- [1.38] X. Meng, P. Sewell, S. Phang, A. Vukovic, and T. M. Benson, “Modeling Curved Carbon Fiber Composite (CFC) Structures in the Transmission-Line

- Modeling (TLM) Method,” *IEEE Trans. Electromagn. Compat.*, vol. 57, no. 3, pp. 384–390, 2015.
- [1.39] A. Elkalsh, A. Vukovic, P. D. Sewell, and T. M. Benson, “Electro-thermal modelling for plasmonic structures in the TLM method,” *Opt. Quantum Electron.*, vol. 48, no. 4, p. 263, 2016.
- [1.40] J. Paul, C. Christopoulos, and D. Thomas, “Perfectly matched layer for transmission line modelling (TLM) method,” *Electron. Lett.*, vol. 33, no. 9, p. 729, 1997.
- [1.41] N. Pena and M. Ney, “Absorbing-boundary conditions using perfectly matched-layer (PML) technique for three-dimensional TLM simulations,” *IEEE Trans. Microw. Theory Tech.*, vol. 45, no. 10, pp. 1749–1755, 1997.
- [1.42] J. L. Dubard and D. Pompei, “Optimization of the PML Efficiency in 3-D TLM Method,” *IEEE Trans. Microw. Theory Tech.*, vol. 48, no. 7 PART 1, pp. 1081–1088, 2000.
- [1.43] J.-P. Berenger, “Perfectly matched layer for the FDTD solution of wave-structure interaction problems,” *IEEE Trans. Antennas Propag.*, vol. 44, no. 1, pp. 110–117, 1996.
- [1.44] J. A. Roden and S. D. Gedney, “Convolution PML (CPML): An efficient FDTD implementation of the CFS-PML for arbitrary media,” *Microw. Opt. Technol. Lett.*, vol. 27, no. 5, pp. 334–339, dec 2000.
- [1.45] A. E. Yilmaz, D. S. Weile, B. Shanker, J. M. Jin, and E. Michielssen, “Fast analysis of transient scattering in lossy media,” *IEEE Antennas Wirel. Propag. Lett.*, vol. 1, no. 1, pp. 14–17, 2002.
- [1.46] H. Baci, A. E. Yilmaz, V. Lomakin, and E. Michielssen, “Fast solution of mixed-potential time-domain integral equations for half-space environments,” *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 2, pp. 269–279, 2005.
- [1.47] P. Jiang and E. Michielssen, “Multilevel Plane Wave Time Domain-Enhanced MOT Solver for Analyzing Electromagnetic Scattering from Objects Residing in Lossy Media,” in *2005 IEEE Antennas Propag. Soc. Int. Symp.*, vol. 3B, no. 2. IEEE, 2005, pp. 447–450.
- [1.48] Pei-Lin Jiang and E. Michielssen, “Multilevel PWTD-Enhanced CFIE Solver for Analyzing EM Scattering from PEC Objects Residing in Lossy Media,” in *2006 IEEE Antennas Propag. Soc. Int. Symp.*, vol. 11270, no. 2004. IEEE, 2006, pp. 2967–2970.
- [1.49] A. POGGIO and E. MILLER, *Computer Techniques for Electromagnetics*. Elsevier, 1973.

Electromagnetic Theory

In this chapter, the theory of electromagnetism and Maxwell's equations will be introduced. The theory will be derived from first principles and result in the frequency domain representation formulas. This will give a solid foundation for which to base all proceeding time domain theory.



2.1 Maxwell's Equations

The fundamental equations that govern the propagation of electromagnetic waves are Maxwell's equations. These equations consist of Faraday's law of induction (2.1), the Ampere-Maxwell law (2.2), Gauss' law for magnetism (2.3), Gauss' law (2.4), and the continuity equation (2.5). The constitutive equations, 2.6 and 2.7, are shown for a linear isotropic medium. The description of each symbol is shown in table 2.1, where a bold variable indicates a vector quantity in space and time. Derivation and further explanation for

these equations can be found in reference [2.1].

$$\nabla \times \mathbf{e} = -\partial_t \mathbf{b} - \mathbf{m} \quad (2.1)$$

$$\nabla \times \mathbf{h} = \partial_t \mathbf{d} + \mathbf{j} \quad (2.2)$$

$$\nabla \cdot \mathbf{b} = \rho_m \quad (2.3)$$

$$\nabla \cdot \mathbf{d} = \rho_e \quad (2.4)$$

$$\nabla \cdot \mathbf{j} = -\partial_t \rho_e \quad (2.5)$$

$$\mathbf{b} = \mu \mathbf{h} \quad (2.6)$$

$$\mathbf{d} = \varepsilon \mathbf{e} \quad (2.7)$$

∂_t denotes the differential with respect to time, $\nabla \cdot$ indicates a divergence operation, and $\nabla \times$ indicates a curl operation which can be explicitly be written as shown in equations 2.8 and 2.9 respectively.

$$\nabla \cdot \boldsymbol{\varphi} \equiv \frac{\partial \varphi_x}{\partial x} + \frac{\partial \varphi_y}{\partial y} + \frac{\partial \varphi_z}{\partial z} \quad (2.8)$$

$$\nabla \times \boldsymbol{\varphi} \equiv \begin{vmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ \varphi_x & \varphi_y & \varphi_z \end{vmatrix} \quad (2.9)$$

Material parameters are chosen so that they are assumed to be linear and non-complex, as shown in equations 2.10 and 2.11, where the free space parameter values are shown in 2.12 and 2.13.

$$\varepsilon = \varepsilon_r \varepsilon_0 \quad (2.10)$$

$$\mu = \mu_r \mu_0 \quad (2.11)$$

$$\varepsilon_0 = 8.854187817 \times 10^{-12} \text{ Fm}^{-1} \quad (2.12)$$

$$\mu_0 = 4\pi \times 10^{-7} \text{ Hm}^{-1} \quad (2.13)$$

Table 2.1: Description of notations related to electromagnetic problems

Symbol	Unit	Name
\mathbf{e}	V/m	Electric field intensity
\mathbf{h}	A/m	Magnetic field intensity
\mathbf{d}	C/m ²	Electric flux density
\mathbf{b}	Wb/m ²	Magnetic flux density
\mathbf{j}	V/m ²	Electric current density
\mathbf{m}	A/m ²	Magnetic current density
ρ_e	C	Electric charge
ρ_m	Wb	Magnetic charge
ϵ	F/m	Electric permittivity
ϵ_0	F/m	Free-space permittivity
ϵ_r	-	Relative permittivity
μ	H/m	Magnetic permeability
μ_0	H/m	Free-space permeability
μ_r	-	Relative permeability

2.2 Representation Formulas

We can reduce equations 2.1-2.7 to

$$\begin{aligned}\nabla \times \mathbf{e} &= -\partial_t \mu \mathbf{h} - \mathbf{m} \\ \nabla \times \mathbf{h} &= \partial_t \epsilon \mathbf{e} + \mathbf{j}\end{aligned}\tag{2.14}$$

We can then combine the equations in 2.14 to obtain the scalar wave equations,

$$\begin{aligned}\nabla^2 \mathbf{e} - \frac{1}{c^2} \partial_t^2 \mathbf{e} &= -\partial_t \mu \mathbf{j} - \nabla \times \mathbf{m} \\ \nabla^2 \mathbf{h} - \frac{1}{c^2} \partial_t^2 \mathbf{h} &= -\partial_t \epsilon \mathbf{m} + \nabla \times \mathbf{j}\end{aligned}\tag{2.15}$$

where $c(= 1/\sqrt{\epsilon\mu})$ is the speed of propagation. The theory henceforth will consider problems in the absence of magnetic sources.

The solution for both the electric and magnetic fields in 2.15 is fulfilled by the scalar and vector potentials after imposing the Lorenz gauge condition. The solution can be found using the Green's function with a Dirac pulse excitation acting instantaneously at $t = t'$ in time and $\mathbf{r} - \mathbf{r}'$ in space,

$$\nabla^2 G(R, t) - \frac{1}{c^2} \partial_t^2 G(R, t) = -\delta(R) \delta(t - t') \quad (2.16)$$

where the distance from the origin is $R = |\mathbf{r} - \mathbf{r}'|$, and G denotes the Green's function, which is explained in more detail in section 2.2.1. In 3D space, the distance is computed using Cartesian coordinates by $R = \sqrt{x^2 + y^2 + z^2}$, in 2D space this is $R = \sqrt{x^2 + y^2}$, and in 1D space this is simply $R = x$. The Dirac delta function can be loosely defined as

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

however the Dirac delta is not technically a function and can be more accurately defined as a distribution [2.1]. If $f(x)$ is a continuous function, the Dirac delta function has the ability to pick out the value of $f(x)$ at $x = 0$:

$$\langle \delta, f \rangle = \int_{-\infty}^{\infty} f(x) \delta(x - a) dx = f(a)$$

By convolving 2.15 with the Green's function and 2.16 with the field, and then integrating the difference over the entire exterior region, Ω_∞ , we obtain 2.17, where the incident field is denoted by \mathbf{u}^i . \mathbf{r}' and \mathbf{r} denote the source and observation locations respectively. We shall use \mathbf{u} as a symbol to represent the vector field which will enable us to derive the TE and TM modes simultaneously.

$$\int_{\Omega_\infty} G * \nabla^2 \mathbf{u}(\mathbf{r}, t) - \mathbf{u}(\mathbf{r}, t) * \nabla^2 G d\Omega = -\mathbf{u}^i(\mathbf{r}', t) + \int_{\Omega_\infty} \mathbf{u}(\mathbf{r}', t) \delta d\Omega \quad (2.17)$$

where $*$ indicates a temporal convolution which is defined as

$$f(t) * g(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$

The second scalar Green's theorem states

$$\int_{\Omega_C} \psi \nabla^2 \varphi = \int_{\Gamma} \psi \nabla_n \varphi \, dS - \int_{\Omega} \varphi \nabla_n \psi \, dS \quad (2.18)$$

where Γ is the surface of the volume, Ω , and the normal unit vector $\hat{\mathbf{n}}$ points outwards from the surface. The spatial derivative in the direction of the normal unit vector can also be written as

$$\nabla_n \varphi = \frac{\partial \varphi}{\partial n} = \hat{\mathbf{n}} \cdot \nabla \varphi$$

We then apply the second scalar Green's theorem to the left hand side (LHS) of 2.17 and the definition of the Dirac delta function to the right hand side (RHS) to get

$$\begin{aligned} \oint_{\Gamma_{\infty}} \left[G^* \frac{\partial(\mathbf{r}, t)}{\partial n} - \mathbf{u}(\mathbf{r}, t)^* \frac{\partial G}{\partial n} \right] d\Gamma + \oint_{\Gamma_0} \left[\mathbf{u}(\mathbf{r}, t)^* \frac{\partial G}{\partial n} - G^* \frac{\partial \mathbf{u}(\mathbf{r}, t)}{\partial n} \right] d\Gamma \\ + \mathbf{u}^i(\mathbf{r}, t) = \mathbf{u}(\mathbf{r}', t) \end{aligned} \quad (2.19)$$

where Γ_0 is the boundary of the object, and Γ_{∞} is the boundary of the exterior region (a circle with radius approaching infinity). The radiation condition is satisfied by both G and \mathbf{u} which, when substituted into 2.19, makes the boundary integral over Γ_{∞} tend to zero as it's radius approaches infinity, thus the total field becomes

$$\mathbf{u}(\mathbf{r}', t) = \mathbf{u}^i(\mathbf{r}', t) + \underbrace{\oint_{\Gamma_0} \left[\mathbf{u}(\mathbf{r}, t)^* \frac{\partial G}{\partial n} - G^* \frac{\partial \mathbf{u}(\mathbf{r}, t)}{\partial n} \right] d\Gamma}_{\mathbf{u}^s(\mathbf{r}', t)} \quad (2.20)$$

where \mathbf{u}^s is the scattered field.

Equation 2.20 states that once the field and its normal derivative are known on the boundary of a domain, the field everywhere in that domain can be found.

It is customary in literature at this stage to exchange source and observation points, resulting in the scattered field shown in 2.21. We may also want to take the normal derivative of the scattered field when the observation point is

close to the surface of the scatterer as shown in 2.22, where the unit normal vector faces inwards for a point inside the boundary.

$$\mathbf{u}^s(\mathbf{r}, t) = \oint_{\Gamma_0} \left[\mathbf{u}(\mathbf{r}', t) * \frac{\partial G(\mathbf{r}, \mathbf{r}', t)}{\partial n'} - G(\mathbf{r}, \mathbf{r}', t) * \frac{\partial \mathbf{u}(\mathbf{r}', t)}{\partial n'} \right] d\Gamma' \quad (2.21)$$

$$\frac{\partial \mathbf{u}^s(\mathbf{r}, t)}{\partial n} = \oint_{\Gamma_0} \left[\mathbf{u}(\mathbf{r}', t) * \frac{\partial^2 G(\mathbf{r}, \mathbf{r}', t)}{\partial n \partial n'} - \frac{\partial G(\mathbf{r}, \mathbf{r}', t)}{\partial n} * \frac{\partial \mathbf{u}(\mathbf{r}', t)}{\partial n'} \right] d\Gamma' \quad (2.22)$$

2.2.1 The Green's Function

Mathematically, the Green's function is the kernel of an integral operator that represents the inverse of a differential operator. Physically, it is the response of a system when a unit point source is applied to the system [2.2].

2.2.1.1 Green's Function in 2D

Since the 2D Green's function can be interpreted as an infinitely long line source in 3D, we can derive it by integrating the 3D Green's function along the z axis.

The well-known causal 3D Green's function in the time domain is

$$g_{3D}(R, t) = \frac{\delta(t - R/c)}{4\pi R} \quad (2.23)$$

A rigorous derivation for this function can be found in many textbooks on function analysis, for example in references [2.2,2.3].

The integral to solve in order to obtain the 2D Green's function is therefore

$$g_{2D}(R, t) = \frac{1}{4\pi} \int_{-\infty}^{\infty} \frac{\delta(t - \sqrt{x^2 + y^2 + z^2}/c)}{\sqrt{x^2 + y^2 + z^2}} dz \quad (2.24)$$

By a change of variables, the generalized scaling property may be written

as

$$\int_{-\infty}^{\infty} f(x) \delta(g(x)) dx = \sum_i \frac{f(x_i)}{|g'(x_i)|} \quad (2.25)$$

where the summation extends over all roots of $g(z)$. Between $[-\infty, \infty]$, these roots are

$$z = \pm \sqrt{(ct)^2 - (x^2 + y^2)} \quad (2.26)$$

By applying 2.25 to 2.24, and then substituting the two roots from 2.26, we can obtain

$$g_{2D}(R, t) = \frac{1}{4\pi} \frac{2c}{\sqrt{(ct)^2 - R^2}} \quad (2.27)$$

The real roots occur when $ct > R$, which then introduces the Heaviside function, denoted H . The Heaviside function is also known as the unit step function and can be defined as

$$H(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0, \end{cases}$$

It can also be differentiated as

$$\frac{d}{dx} H(x) = \delta(x)$$

The causal 2D TD Green's function can therefore be written as

$$g_{2D}(R, t) = \frac{H(t - R/c)}{2\pi \sqrt{t^2 - (R/c)^2}} \quad (2.28)$$

The 2D Green's function can be interpreted as the field radiated in the x, y plane by an infinitely long line source in the z direction. This gives rise to a singularity at $t = R/c$, and also an exponential decaying signal which follows the initial disturbance set by the source.

2.2.1.2 Green's Function in 1D

To find the time domain Green's function in 1D, we first derive the frequency domain version. The FD Green's function is the solution to the Helmholtz equation shown in 2.16, which for the 1D frequency domain is

$$\partial_R^2 G_{1D}(R) - k^2 G_{1D}(R) = -\delta(R) \quad (2.29)$$

where k is the wavenumber. The solution can be found to be

$$G_{1D}(R) = Ae^{-ikR} \quad (2.30)$$

where A is a coefficient which can be found by integrating 2.16 over the origin (the singularity) between an infinitesimal distance $[-\epsilon, \epsilon]$, thus giving

$$G_{1D}(R) = \frac{1}{2ik} e^{-ikR} \quad (2.31)$$

The TD Green's function is expressed as the field radiated by a Dirac source, and can be found by applying the inverse Fourier transform to 2.31. The inverse Fourier transform is defined as:

$$\mathcal{F}^{-1}g(x) := \int_{\mathbb{R}^n} e^{2\pi i x \cdot \xi} g(\xi) d\xi$$

Consequently, we can obtain the 1D Green's function in the time domain:

$$g_{1D}(R, t) = \frac{1}{2} H(ct - R) \quad (2.32)$$

There are many other ways to derive this formula, either using the method of d'Alembert to find the solution to the 1D wave equation, or by integrating the 2D Green's function along the y axis.

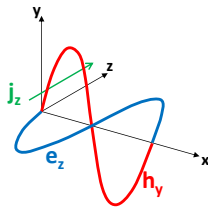


Figure 2.1: 1D electromagnetic wave.

2.2.2 1D Representation Formulas

For an x-directed, one-dimensional electromagnetic wave emanating from an electric current source in the absence of magnetic current as shown in figure 2.1, we can combine the equations in 2.21 and 2.22 to obtain

$$\begin{pmatrix} \mathbf{u}^s(\mathbf{r}, t) \\ \partial_n \mathbf{u}^s(\mathbf{r}, t) \end{pmatrix} = \begin{pmatrix} \int \frac{\partial G_{1D}(\mathbf{r}, \mathbf{r}')}{\partial n'} dr' & - \int G_{1D}(\mathbf{r}, \mathbf{r}') dr' \\ \int \frac{\partial^2 G_{1D}(\mathbf{r}, \mathbf{r}')}{\partial n \partial n'} dr' & - \int \frac{\partial G_{1D}(\mathbf{r}, \mathbf{r}')}{\partial n} dr' \end{pmatrix} * \begin{pmatrix} \mathbf{u}(\mathbf{r}') \\ \partial_n \mathbf{u}(\mathbf{r}') \end{pmatrix} \quad (2.33)$$

where the spatial integration occurs over the surface of a 1D domain, i.e. a single point, which is equivalent to evaluating at that point, thus can be excluded in the 1D case.

We must note that the normal derivatives are linked using the equations in 2.14. For example, if $\mathbf{u} = \mathbf{e}$, the normal derivative is derived as shown in 2.34.

$$\begin{aligned} \frac{\partial \mathbf{e}}{\partial n} &= \hat{\mathbf{n}} \cdot \nabla \mathbf{e} \\ &= (\hat{\mathbf{t}} \times \hat{\mathbf{z}}) \cdot \frac{\partial \mathbf{e}}{\partial x} \\ &= -\hat{\mathbf{t}} \cdot \left(\frac{\partial \mathbf{e}}{\partial x} \times \hat{\mathbf{z}} \right) \\ &= -\mu (\hat{\mathbf{t}} \cdot \hat{\mathbf{y}}) \partial_t \mathbf{h} \end{aligned} \quad (2.34)$$

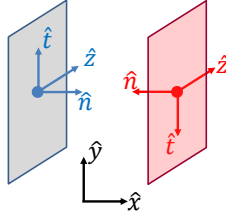


Figure 2.2: Local normal unit vectors on opposing faces.

where $\hat{\mathbf{t}}$ is the tangential unit vector, and $\hat{\mathbf{y}}$ is the unit vector in the y-direction. The dot product, $(\hat{\mathbf{t}} \cdot \hat{\mathbf{y}})$, can be either $+1$ for a wave travelling in the positive direction, or -1 for a wave travelling in the negative direction, as depicted in figure 2.2.

Substituting 2.34 into 2.33, we get the 1D representation formula for the transverse magnetic (TM) case, where the transverse electric (TE) case is identical except for the orthogonal polarity and change in material parameter ($\mu \rightarrow \varepsilon$)

$$\begin{pmatrix} e_z^s(\mathbf{r}, t) \\ h_t^s(\mathbf{r}, t) \end{pmatrix} = \begin{pmatrix} \frac{\partial G_{1D}(R, t)}{\partial n'} & \mu \frac{\partial G_{1D}(R, t)}{\partial t} \\ -\frac{1}{\mu} \int_t \frac{\partial^2 G_{1D}(R, t)}{\partial n \partial n'} dt & -\frac{\partial G_{1D}(R, t)}{\partial n} \end{pmatrix} * \begin{pmatrix} e_z(\mathbf{r}', t) \\ h_t(\mathbf{r}', t) \end{pmatrix} \quad (2.35)$$

where distance $R = |\mathbf{r} - \mathbf{r}'|$, and the tangential components of the electric and magnetic fields are e_z and $h_t (= \hat{\mathbf{t}} \cdot \mathbf{h}_{xy})$ respectively.

As can be seen in 2.35, the spatial and temporal derivatives, as well as temporal integral, are required of the TD Green's function shown in 2.32. These can be found as

$$\begin{aligned} \frac{\partial g_{1D}(R, t)}{\partial t} &= \frac{c}{2} \delta(ct - R) \\ \frac{\partial g_{1D}(R, t)}{\partial n'} &= -\frac{1}{2} \delta(ct - R) \\ \frac{\partial g_{1D}(R, t)}{\partial n} &= \frac{1}{2} (\hat{\mathbf{n}} \cdot \hat{\mathbf{n}}') \delta(ct - R) \\ \int_t \frac{\partial^2 g_{1D}(R, t)}{\partial n \partial n'} dt &= -\frac{1}{2c} (\hat{\mathbf{n}} \cdot \hat{\mathbf{n}}') \delta(ct - R) \end{aligned} \quad (2.36)$$

Using the Green's function definitions in 2.36, we can now expand 2.35 to give

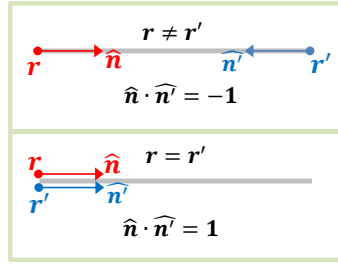


Figure 2.3: Unit normal vectors in 1D for self-patch and otherwise.

the total field

$$\begin{pmatrix} e_z(\mathbf{r}, t) \\ h_t(\mathbf{r}, t) \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\delta(ct - R) & \frac{\eta}{2}\delta(ct - R) \\ \frac{1}{2\eta}(\hat{\mathbf{n}} \cdot \hat{\mathbf{n}}')\delta(ct - R) & \frac{1}{2}(\hat{\mathbf{n}} \cdot \hat{\mathbf{n}}')\delta(ct - R) \end{pmatrix} * \begin{pmatrix} e_z(\mathbf{r}', t) \\ h_t(\mathbf{r}', t) \end{pmatrix} + \begin{pmatrix} e_z^i(\mathbf{r}, t) \\ h_t^i(\mathbf{r}, t) \end{pmatrix} \quad (2.37)$$

In 1D, the dot product of the primed and unprimed unit vectors, $(\hat{\mathbf{n}} \cdot \hat{\mathbf{n}}')$, can be either +1 or -1, and are related as shown in figure 2.3.

The equation in 2.37 is related and used in both 1D TLM and 1D BEM, thus will be revisited in section 6.1.1.

2.2.3 2D Representation Formulas

A cross section of the z -plane (at $z = 0$) of a 3D object is equivalent to a 2D object. The 2D representation formulas as shown in equations 2.21 and 2.22 will hold for any observed point apart from on the surface itself since the 2D Green's function (as shown in 2.28) and its derivative are singular when $\mathbf{r} = \mathbf{r}'$. This is dealt with by deforming the boundary around the observation point and splitting the integral into 2 parts as shown in equation 2.38. [2.4, p.407]

$$\oint_{\Gamma_0} f(\mathbf{r}') d\Gamma' = \lim_{\epsilon \rightarrow 0} \left\{ \int_{\Gamma_0 - \Gamma_\epsilon} f(\mathbf{r}') d\Gamma' + \int_{\Gamma_\epsilon} f(\mathbf{r}') d\Gamma' \right\} \quad (2.38)$$

where Γ_0 is the object surface, and Γ_ϵ is a circular surface surrounding the observation point which has a radius tending to zero, as shown in figure

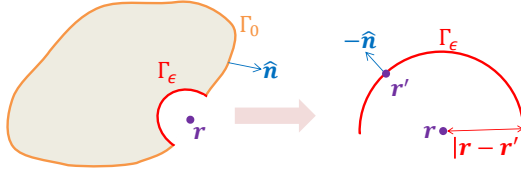


Figure 2.4: Object boundary deformed to exclude singular point.

2.4.

The first integral on the right hand side (RHS) of 2.38 can be denoted as the Cauchy integral which is an integral along Γ_0 but with the singular point excluded. The second integral on the RHS of 2.38 can be solved by evaluating over the boundary of a semi-circle of radius equal to ϵ . Applying this integration to equations 2.21 and 2.22, we obtain

$$\oint_{\Gamma_\epsilon} \left[\mathbf{u}(\mathbf{r}', t) \frac{\partial G(\mathbf{r}, \mathbf{r}', t)}{\partial n'} - G(\mathbf{r}, \mathbf{r}', t) \frac{\partial \mathbf{u}(\mathbf{r}', t)}{\partial n'} \right] d\Gamma' = \frac{1}{2} \mathbf{u}(\mathbf{r}', t) \quad (2.39)$$

$$\oint_{\Gamma_\epsilon} \left[\mathbf{u}(\mathbf{r}', t) \frac{\partial^2 G(\mathbf{r}, \mathbf{r}', t)}{\partial n \partial n'} - \frac{\partial G(\mathbf{r}, \mathbf{r}', t)}{\partial n} \frac{\partial \mathbf{u}(\mathbf{r}', t)}{\partial n'} \right] d\Gamma' = \frac{1}{2} \frac{\partial \mathbf{u}(\mathbf{r}', t)}{\partial n'} \quad (2.40)$$

Thus using Cauchy principle value integration, the representation formulas 2.21 and 2.22 become

$$\mathbf{u}^s(\mathbf{r}, t) = \frac{1}{2} \mathbf{u}(\mathbf{r}', t) \quad (2.41)$$

$$+ p.v. \int_{\Gamma_0} \left[\mathbf{u}(\mathbf{r}', t) \frac{\partial G(\mathbf{r}, \mathbf{r}', t)}{\partial n'} - G(\mathbf{r}, \mathbf{r}', t) \frac{\partial \mathbf{u}(\mathbf{r}', t)}{\partial n'} \right] d\Gamma' \quad (2.42)$$

$$\frac{\partial \mathbf{u}^s(\mathbf{r}, t)}{\partial n} = \frac{1}{2} \frac{\partial \mathbf{u}(\mathbf{r}', t)}{\partial n'} \quad (2.43)$$

$$+ p.v. \oint_{\Gamma_0} \left[\mathbf{u}(\mathbf{r}', t) \frac{\partial^2 G(\mathbf{r}, \mathbf{r}', t)}{\partial n \partial n'} - \frac{\partial G(\mathbf{r}, \mathbf{r}', t)}{\partial n} \frac{\partial \mathbf{u}(\mathbf{r}', t)}{\partial n'} \right] d\Gamma' \quad (2.44)$$

where *p.v.* stands for principle value.

From now on, principle value integration is assumed whenever the observed

point is on the boundary. Equations 2.42 and 2.44 can be written more concisely as

$$\begin{pmatrix} \mathbf{u}^s \\ \partial_n \mathbf{u}^s \end{pmatrix} = \begin{pmatrix} \frac{1}{2} + \int_{\Gamma} \frac{\partial G(\mathbf{r}, \mathbf{r}', t)}{\partial n'} \varphi(\mathbf{r}') dr' & - \int_{\Gamma} G(\mathbf{r}, \mathbf{r}', t) \varphi(\mathbf{r}', t) dr' \\ \int_{\Gamma} \frac{\partial^2 G(\mathbf{r}, \mathbf{r}', t)}{\partial n \partial n'} \varphi(\mathbf{r}') dr' & \frac{1}{2} - \int_{\Gamma} \frac{\partial G(\mathbf{r}, \mathbf{r}', t)}{\partial n} \varphi(\mathbf{r}') dr' \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \partial_n \mathbf{u} \end{pmatrix} \quad (2.45)$$

The transverse and longitudinal components of u are found by taking the tangential component of equations 2.1 and 2.2, then by taking the vector triple product, we obtain

$$\partial_n e_z = \partial_t \mu (\hat{\mathbf{n}} \times \mathbf{h}_{xy}) \quad (2.46)$$

$$\partial_n h_z = -\partial_t \varepsilon (\hat{\mathbf{n}} \times \mathbf{e}_{xy}) \quad (2.47)$$

The normal derivatives do not have to be continuous at the interface between two homogeneous, isotropic materials but can be transformed into tangential derivatives that are continuous by application in an orthogonal axis set $(\hat{\mathbf{n}}, \hat{\mathbf{t}}, \hat{\mathbf{z}})$. [2.5, p.11]

Substituting 2.46 and 2.47 into 2.45, we get the time domain representation formulas for the TM and TE cases, respectively, as shown in equations 2.48 and 2.49.

$$\begin{pmatrix} e_z \\ h_t \end{pmatrix} = \begin{pmatrix} \frac{1}{2} + D & -\mu S \\ -\frac{N}{\mu} & \frac{1}{2} - D' \end{pmatrix} \begin{pmatrix} e_z \\ h_t \end{pmatrix} + \begin{pmatrix} e_z^i \\ h_t^i \end{pmatrix} \quad (2.48)$$

$$\begin{pmatrix} h_z \\ e_t \end{pmatrix} = \begin{pmatrix} \frac{1}{2} + D & \varepsilon S \\ \frac{N}{\varepsilon} & \frac{1}{2} - D' \end{pmatrix} \begin{pmatrix} h_z \\ e_t \end{pmatrix} + \begin{pmatrix} h_z^i \\ e_t^i \end{pmatrix} \quad (2.49)$$

where the operators are defined as

$$\begin{aligned}
 D\varphi(\mathbf{r}', t) &= \int_{\Gamma} \frac{\partial g(R, t)}{\partial n'} * \varphi(\mathbf{r}', t) \, dr' \\
 D'\varphi(\mathbf{r}', t) &= \int_{\Gamma} \frac{\partial g(R, t)}{\partial n} * \varphi(\mathbf{r}', t) \, dr' \\
 S\varphi(\mathbf{r}', t) &= \int_{\Gamma} g(R, t) * \frac{\partial}{\partial t} \varphi(\mathbf{r}', t) \, dr' \\
 N\varphi(\mathbf{r}', t) &= \int_{\Gamma} \int_t \frac{\partial^2 g(R, t)}{\partial n \partial n'} \varphi(\mathbf{r}', t) \, dt \, dr'
 \end{aligned} \tag{2.50}$$

References

- [2.1] D. Griffiths, *Introduction to Electrodynamics*, 3rd ed. Pearson, 2013.
- [2.2] J. V. Bladel, *Electromagnetic Fields*, 2nd ed., ser. IEEE Series on Electromagnetic Wave Theory, IEEE, Ed. Wiley, 2007.
- [2.3] D. H. Griffel, *Applied functional analysis*. Courier Corporation, 2002.
- [2.4] J.-M. J. Jin, *Theory and Computation of Electromagnetic Fields*, I. PRESS, Ed. John Wiley & Sons, 2011.
- [2.5] J. Fostier, “Parallel Techniques for Fast Multipole Algorithms,” Ph.D. dissertation, Ghent University, 2009.

The Boundary Element Method

The boundary element method solves either the electric, magnetic, or combined field integral equation (EFIE, MFIE, CFIE) for electric/magnetic currents on the surface of an object, which consequently leads to the knowledge of EM fields radiated from the object. BEM can be applied in the frequency domain or the time domain. BEM in the time domain has the advantage of simulating transient responses but also frequency responses can still be acquired through Fourier analysis. The principles of 2D BEM algorithms that have been implemented thus far are explained below. Implementation to allow generic basis and temporal functions will be thoroughly reviewed, and validation and stability for the algorithm will also be discussed.



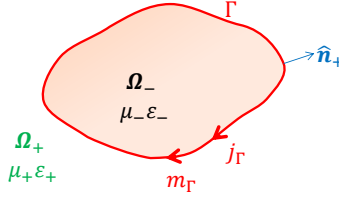


Figure 3.1: Jump conditions at the boundary.

3.1 Jump conditions

A volume (domain Ω_-) is separated from free space (Ω_+) with a closed surface Γ as shown in figure 3.1.

At the surface, the permittivity jumps from ϵ_- to ϵ_+ (along with the permeability) hence the term *jump conditions*. The singular contributions to Maxwell's equations at a point on the surface are derived from equations 2.3, 2.4 and 2.14

$$\begin{aligned}
 [\hat{n} \times \mathbf{e}] &= \hat{n} \times \mathbf{e}_+ - \hat{n} \times \mathbf{e}_- = -\mathbf{m}_\Gamma \\
 [\hat{n} \times \mathbf{h}] &= \hat{n} \times \mathbf{h}_+ - \hat{n} \times \mathbf{h}_- = \mathbf{j}_\Gamma \\
 [\hat{n} \cdot \mathbf{b}] &= \hat{n} \cdot \mathbf{b}_+ - \hat{n} \cdot \mathbf{b}_- = \rho_m \\
 [\hat{n} \cdot \mathbf{d}] &= \hat{n} \cdot \mathbf{d}_+ - \hat{n} \cdot \mathbf{d}_- = \rho_e
 \end{aligned} \tag{3.1}$$

where $\hat{n} \cdot$ signifies the normal components of inductions, and $\hat{n} \times$ signifies the tangential components of fields.

The electric and magnetic current density on the surface, denoted m_Γ and j_Γ respectively, and also the electric and magnetic charge, denoted ρ_e and ρ_m respectively, are the only terms that can be concentrated on the boundary.

If there is no surface charge or surface current, then the right hand values will be zero and the relevant fields are continuous across the interface. These equations, as well as the radiation condition which stipulates fields must radiate outwards, are used to find out if a given set of fields are the correct solutions to Maxwell's equations.

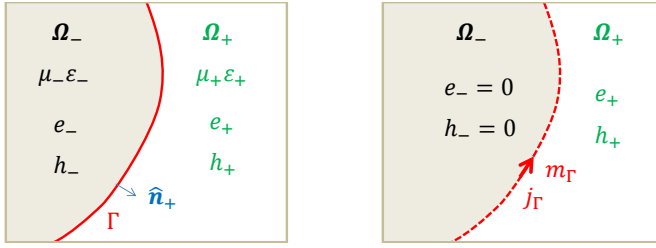


Figure 3.2: Equivalence principle a) before, and b) after, where the fields inside the scatterer are zero and equivalent surface currents are now to be solved.

3.2 Equivalence Principle

In general, the representation formula, which solves the total field everywhere, can be written as:

$$\begin{aligned} \mathbf{u}(\mathbf{r}', t) &= \mathbf{u}^i(\mathbf{r}', t) + \mathbf{u}^s(\mathbf{r}', t) \\ &= \mathbf{u}^i(\mathbf{r}', t) + \int_t \int_{\Gamma} G(R, t) \cdot \boldsymbol{\varphi}_{\Gamma}(\mathbf{r}, t - \tau) d\mathbf{r}' d\tau \end{aligned} \quad (3.2)$$

where the subscripts i and s denote the incident and scattered fields respectively. The integral acts on the whole surface Γ and the Green's function G translates the current density seen at the source point \mathbf{r}' to the field at the observation point \mathbf{r} . The Green's function changes depending on the medium it is associated with. For simplification, we proceed to replace the scatterer with free space and introduce equivalence currents just outside region. This makes the surface currents easier to find since we can now just use the free space Green's function. The solution will still be correct as Maxwell's equations, jump conditions, and radiation conditions are still satisfied.

Consider the interface between an arbitrary material as shown in 3.2a. The fields generated by the material can be generated in the same way using equivalent surface currents as shown in 3.2b.

Using 3.1, we can obtain the relationship between the equivalent currents and

Table 3.1: PEC equations

	EFIE	MFIE
TM	$e_z^i = \mu S j_z$	$\hat{\mathbf{n}} \times \mathbf{h}_{xy}^i = \left(\frac{1}{2} + D'\right) j_z$
TE	$\hat{\mathbf{n}} \times \mathbf{e}_{xy}^i = \frac{N}{\varepsilon} j_{xy}$	$h_z^i = \left(-\frac{1}{2} + D\right) j_{xy}$

the tangential fields on the boundary

$$\begin{aligned}
 \mathbf{j}_\Gamma &= \hat{\mathbf{n}} \times \mathbf{h}_- \\
 &= -\hat{\mathbf{n}} \times \mathbf{h}_+ \\
 \mathbf{m}_\Gamma &= -\hat{\mathbf{n}} \times \mathbf{e}_- \\
 &= \hat{\mathbf{n}} \times \mathbf{e}_+
 \end{aligned} \tag{3.3}$$

3.3 Scattering by a Perfect Electric Conductor

Inside a Perfect Electric Conductor (PEC), the fields are zero; only the surface will sustain electrical surface current, though not magnetic surface current, thus the occurrence of jump conditions at the object boundary gives us the equations in 3.4 for points just outside the surface. [3.1, pp.408-409]

$$\begin{aligned}
 \hat{\mathbf{n}} \times \mathbf{e}_{xy} &= \hat{\mathbf{n}} \times \mathbf{e}_{xy}^i + \hat{\mathbf{n}} \times \mathbf{e}_{xy}^s = m_z = 0 \\
 \hat{\mathbf{n}} \times \mathbf{h}_{xy} &= \hat{\mathbf{n}} \times \mathbf{h}_{xy}^i + \hat{\mathbf{n}} \times \mathbf{h}_{xy}^s = j_z \\
 e_z &= m_{xy} = 0 \\
 h_z \hat{\mathbf{t}} &= -(\hat{\mathbf{n}} \times h_z) \hat{\mathbf{z}} = -j_{xy}
 \end{aligned} \tag{3.4}$$

Plugging 3.4 into 2.48 and 2.49 results in the equations to obtain the unknown current densities from the corresponding incident fields, as shown in table 3.1.

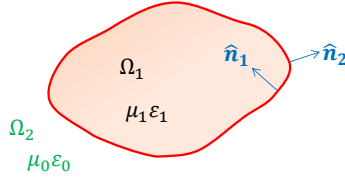


Figure 3.3: An arbitrary penetrable object with labelled regions and normals.

3.4 Scattering by a Penetrable Object

A penetrable object in the context of this thesis is one that can be penetrated by an EM wave.

The scattered field inside and outside of the object needs to be solved when dealing with penetrable objects. This requires two sets of representation formulas, one for each region. Assigning equation 2.48 to region 1 gives equation 3.5, which for brevity will be represented as shown in equation 3.6.

$$\begin{pmatrix} \mathbf{u}_1 \\ \partial_n \mathbf{u}_1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} + D_1 & -\mu_1 S_1 \\ -\frac{1}{\mu_1} N_1 & \frac{1}{2} - D'_1 \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \partial_n \mathbf{u}_1 \end{pmatrix} + \begin{pmatrix} \mathbf{u}_1^i \\ \partial_n \mathbf{u}_1^i \end{pmatrix} \quad (3.5)$$

$$U_1 = Z_1 U_1 + U_1^i \quad (3.6)$$

Similarly, region 2 will have the same form but with subscript 2. Continuity of fluxes normal to the surface is conserved across the boundary and this is the common ground for the representation formula for both regions. However we must note that the normal vector for region 2 points in the opposite direction to that of region 1, as shown in figure 3.3. All terms with relation to a normal unit vector will change as follows

$$\begin{aligned} \partial_n u_2|_{\Omega_2} &= -\partial_n u_2|_{\Omega_1} \\ D_2|_{\Omega_2} &= -D_2|_{\Omega_1} \\ D'_2|_{\Omega_2} &= -D'_2|_{\Omega_1} \end{aligned} \quad (3.7)$$

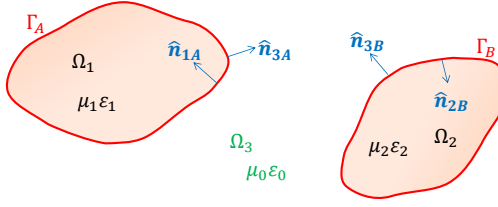


Figure 3.4: Two arbitrary penetrable objects with labelled regions and normals.

Hence the representation formula assigned to region 2 can be written as

$$U_1 = Z'_2 U_1 + U_2^i \quad (3.8)$$

$$Z'_2 = \begin{pmatrix} \frac{1}{2} - D_2 & \mu_2 S_2 \\ \frac{1}{\mu_2} N_2 & \frac{1}{2} + D'_2 \end{pmatrix} \quad (3.9)$$

Taking 3.9 from 3.5 and rearranging to solve for the incident fields yields a Poggio-Miller-Chan-Harrington-Wu-Tsai (PMCHWT) type equation,

$$U_1^i - U_2^i = (Z'_2 - Z_1) U_1 \quad (3.10)$$

$$Z'_2 - Z_1 = \begin{pmatrix} -D_1 - D_2 & \mu_1 S_1 + \mu_2 S_2 \\ \frac{1}{\mu_1} N_1 + \frac{1}{\mu_2} N_2 & D'_1 + D'_2 \end{pmatrix} \quad (3.11)$$

From here we can derive the TM and TE equivalent formulas using 2.46 and 2.47 respectively.

3.5 Scattering by 2 Spatially Distinct Penetrable Objects

When considering 2 spatially distinct penetrable objects, the scattered field will need to be solved for 3 regions. Regions 1 and 2 will denotes the space

inside the objects, and region 3 will denote all space outside these regions. The regions and boundaries are depicted in figure 3.4 where Ω denotes the region, Γ_A denotes the object boundary of Ω_1 , and Γ_B denotes the object boundary of Ω_2 . This requires three sets of representation formulas, one for each region. For brevity, we will represent the equation for regions 1-3 (restricted to boundary A or B) as shown in 3.6 to get

$$U_{1A} = Z_1 U_{1A} + U_{1A}^i \quad (3.12)$$

$$U_{2B} = Z_2 U_{2B} + U_{2B}^i \quad (3.13)$$

$$U_3 = Z_3 U_3 + U_3^i \quad (3.14)$$

where 3.14 contains contributions from both object boundaries, i.e.

$$U_3 = \begin{pmatrix} \mathbf{u}_{3A} \\ \mathbf{u}_{3B} \\ \partial_n \mathbf{u}_{3A} \\ \partial_n \mathbf{u}_{3B} \end{pmatrix} \quad (3.15)$$

Continuity of fluxes normal to the surface is conserved across all boundaries, however we must take into account the change in direction of the normal vector. Taking this into account, equations 3.12 and 3.13 become

$$U_{3A} = Z_1' U_{3A} + U_{1A}^i \quad (3.16)$$

$$U_{3B} = Z_2' U_{3B} + U_{2B}^i \quad (3.17)$$

These equations must be superimposed to match the form of equation 3.14

$$U_3 = Z_{12} U_3 + U_{12}^i \quad (3.18)$$

where

$$Z_{12} = \begin{pmatrix} \frac{1}{2} - D_1 & 0 & \mu_1 S_1 & 0 \\ 0 & \frac{1}{2} - D_2 & 0 & \mu_2 S_2 \\ \frac{1}{\mu_1} N_1 & 0 & \frac{1}{2} + D'_1 & 0 \\ 0 & \frac{1}{\mu_2} N_2 & 0 & \frac{1}{2} + D'_2 \end{pmatrix} \quad U_{12}^i = \begin{pmatrix} \mathbf{u}_{1A}^i \\ \mathbf{u}_{2B}^i \\ \partial_n \mathbf{u}_{1A}^i \\ \partial_n \mathbf{u}_{2B}^i \end{pmatrix} \quad (3.19)$$

Subtract 3.14 from 3.18 and solve for the incident field to obtain the final equation

$$U_3^i - U_{12}^i = (Z_{12} - Z_3)U_3 \quad (3.20)$$

3.6 Reducing the Singularity

To reduce the order of the singularity contained by the hypersingular integrals in the N operator, we apply the Green's identity as derived in [3.2, p.6-7] to produce

$$\begin{aligned} N\varphi(\mathbf{r}', t) &= -\hat{\mathbf{n}} \cdot \nabla \int_{\Gamma} \hat{\mathbf{n}}' \cdot \nabla' G(R, t) \varphi(\mathbf{r}', t) \, dr' \\ &= \int_{\Gamma} \hat{\mathbf{t}} \cdot \nabla^2 G(R, t) \hat{\mathbf{t}}' \varphi(\mathbf{r}', t) - \hat{\mathbf{t}} \cdot \nabla G(R, t) \nabla' \varphi(\mathbf{r}', t) \, dr' \end{aligned} \quad (3.21)$$

where ∇G and $\nabla \cdot G$ is the surface gradient and surface divergence of the Green's function, respectively, which reduce to simple derivatives along the boundary in the counter-clockwise direction. The equation in 3.21 can be reduced using the Helmholtz equation when $\mathbf{r} \neq \mathbf{r}'$, which stipulates $\nabla^2 G = \frac{1}{c^2} \partial_t^2 G$.

We can split the N operator into two components; the first term being the singular contribution, N_s , and the second term being the hypersingular con-

tribution, N_h . Thus 3.21 becomes

$$\begin{aligned} N\varphi(\mathbf{r}', t) &= N_s\varphi(\mathbf{r}', t) + N_h\varphi(\mathbf{r}', t) \\ N_s\varphi(\mathbf{r}', t) &= \frac{1}{c^2} \hat{\mathbf{t}} \cdot \int_{\Gamma} \hat{\mathbf{t}}' \cdot \mathbf{g}(R, t) * \frac{\partial}{\partial t} \varphi(\mathbf{r}', t) dr' \\ N_h\varphi(\mathbf{r}', t) &= -\hat{\mathbf{t}} \cdot \int_{\Gamma} \nabla \mathbf{g}(R, t) * \int_t [\nabla' \varphi(\mathbf{r}', t)] dt dr' \end{aligned} \quad (3.22)$$

3.7 Implementation

We can generalise each operator in equation 2.50 as

$$\mathcal{V}(\mathbf{r}', t) = \mathcal{Z}(R, t) * \varphi(\mathbf{r}, t) \quad (3.23)$$

where \mathcal{V} is the incident field, φ is the unknown boundary field, and \mathcal{Z} is the appropriate operator.

When considering implementation on arbitrary surfaces, meshing must be used to limit the number of solutions along the contour to a finite amount. The length of the longest edge should generally be less than a tenth of the smallest wavelength encountered in the simulation.

The unknown current density must also be discretised using a linear combination of spatial, $S_n(\mathbf{r})$, and temporal, $T(t)$, basis functions as shown in equation 3.24, where \mathbf{r} is a selected point in space, t is a selected time, N_F is the total number of basis functions on the contour, and N_T is the total number of temporal basis functions, a.k.a. number of timesteps.

$$\varphi(\mathbf{r}, t) \cong \sum_{i=0}^{N_T-1} \sum_{n=1}^{N_F} x_{i,n} T(t - t_i) S_n(\mathbf{r}) \quad (3.24)$$

Spatial basis functions are linear distributions on an edge of a geometric shape and are the building blocks for the current density at the source (denoted by primed vectors). Since the Green's function must be integrated over both source and observation points, testing functions must also be used for the observation edges (denoted by unprimed vectors). For symmetry and robustness of the discretised operators, we use the spatial Galerkin method, which stipu-

lates that the basis and testing functions in space are the same. Galerkin-in-time schemes have been recently accomplished [3.3], but for ease of implementation in this project, the temporal testing functions will be Dirac delta/pulse functions.

3.7.1 Testing functions

We can test equation 3.23 over the whole surface using a testing function, $\kappa(\mathbf{r})$:

$$\int_{\Gamma} \kappa(\mathbf{r}) \mathcal{V}(\mathbf{r}', t) dr = \int_{\Gamma} \kappa(\mathbf{r}) \mathcal{Z}(R, t) \varphi(\mathbf{r}, t) dr \quad (3.25)$$

For the special case of the N_h operator, we can use partial integration, i.e.

$$\int_{\Gamma} \hat{\mathbf{t}} \kappa \cdot \varphi = - \int_{\Gamma} \nabla \cdot \hat{\mathbf{t}} \kappa \varphi + \int_{\partial\Gamma} \hat{\mathbf{n}} \cdot \hat{\mathbf{t}} \kappa \varphi$$

however the testing function vectors are always orthogonal to the boundary normals, so $\hat{\mathbf{n}} \cdot \hat{\mathbf{t}} = 0$, thus:

$$\int_{\Gamma} \kappa(\mathbf{r}) N_h \varphi(\mathbf{r}', t) = \int_{\Gamma} \int_{\Gamma} \left[\nabla \cdot \kappa(\mathbf{r}) g(R, t) * \int_t [\nabla' \cdot \varphi(\mathbf{r}', t)] dt \right] dr' dr \quad (3.26)$$

From equation 3.25, we discretise the unknown as shown in 3.24. We then use the Galerkin-in-space scheme which stipulates that the spatial testing functions should equal the basis functions, and then exploit time translation symmetry (using $k = j - i$) to get the discrete convolution:

$$\int_{\Gamma} S_m(\mathbf{r}) \mathcal{V}(\mathbf{r}', t_j) dr = \sum_{k=0}^{j-1} \sum_{n=1}^{N_F} x_{j-k,n} \left(\int_{\Gamma} \int_{\Gamma} S_m(\mathbf{r}) \mathcal{Z}(R, t_k) S_n(\mathbf{r}') dr' dr \right) \quad (3.27)$$

where $t_j = j\Delta t$ and $j = 0, 1, \dots, N_T - 1$.

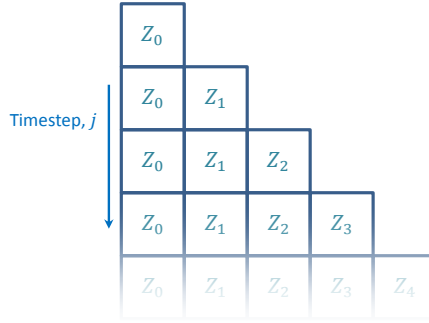


Figure 3.5: Operators used in the marching-on-in-time routine.

We can describe equation 3.27 in matrix form,

$$\{V_m\}_j = \sum_{k=0}^{j-1} [Z_{m,n}]_k \{x_n\}_{j-k} \quad \text{for } j = 0, 1, \dots, N_T - 1 \quad (3.28)$$

where $m, n = 1, 2, \dots, N_F$ denote the current spatial testing and basis functions respectively, k denotes the current temporal basis function, and matrix Z is of size $N_F^2 \times N_T$. For example, one element of matrix Z can be found using:

$$Z_{m,n,k} = \int_{\Gamma} \int_{\Gamma} S_m(\mathbf{r}) \cdot F(R, k) S_n(\mathbf{r}') dr' dr \quad (3.29)$$

where F represents the appropriate convolution depending which operator from 2.50 is being computed.

To solve 3.29, we use the marching-on-in-time (MOT) scheme. For each time step, j , we can deduce the unknown using:

$$\{x\}_j = [Z]_0^{-1} \left(\{V\}_j - \sum_{k=1}^{j-1} [Z]_k \{x\}_{j-k} \right) \quad (3.30)$$

For example, when $j = 0$, $\{x\}_0$ can be solved directly. When $j = 1$, the system contains $\{x\}_1$ and the previously found $\{x\}_0$. This process continues in the same manner up to $j = N_T - 1$ for which the system contains all past values of $\{x\}$, as depicted in Fig. 3.5.

Once x is known, the unknown can then be derived using equation 3.24. This simply equates to $\varphi = x$ for integer values of timestep and vertex and function. This is because both spatial and temporal basis functions are unity at vertex n and timestep i , and 0 elsewhere. For more accuracy between vertices or between time steps, one can simply use linear interpolation.

3.7.2 Piecewise Polynomials

Before continuing, we must introduce the concept of a generic piecewise polynomial; a function split into several sub-functions. Each polynomial sub-function is described by an equation containing coefficients and a single positive and descending variable with integer exponents. The greater the polynomial degree, the more coefficients there will be, e.g. a linear polynomial (of degree 1) will have 2 coefficients, whereas a quadratic polynomial (of degree 2) will have 3 coefficients.

We can evaluate any piecewise polynomial using

$$S(x) = \begin{cases} \sum_{p=0}^P d_{\alpha,p} x^{P-p} & \text{if } x \in \alpha \\ 0, & \text{elsewhere} \end{cases} \quad (3.31)$$

where each polynomial coefficient is denoted by d , the maximum degree is denoted by P , the variable is denoted by x , and $\alpha = 1, 2, \dots, \Upsilon$ denotes a particular sub-function (which acts between partitions). A typical polynomial will have the form $d_{\alpha,0}x^P + d_{\alpha,1}x^{P-1} + \dots + d_{\alpha,P}$. As such, a coefficient matrix has the form

$$\text{Coefficient matrix} = \begin{bmatrix} d_{1,0} & d_{1,1} & \dots & d_{1,P} \\ d_{2,0} & d_{2,1} & \dots & d_{2,P} \\ \vdots & \vdots & \ddots & \vdots \\ d_{\Upsilon,0} & d_{\Upsilon,1} & \dots & d_{\Upsilon,P} \end{bmatrix} \quad (3.32)$$

where the matrix size is $[(P+1) \times \Upsilon]$, and Υ is the number of sub-functions.

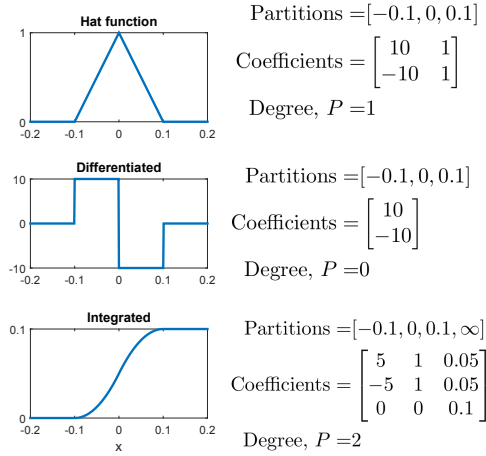


Figure 3.6: Piecewise polynomial example using a hat function which has then be spatially differentiated and integrated.

This concept may be better explained with a simple example. Consider the piecewise linear ‘hat’ function operating in the partitions located at $[-0.1, 0, 0.1]$ with a height of 1 as shown at the top of figure 3.6. The piecewise definition of this function is shown in equation 3.33. Figure 3.6 also shows the differentiated and integrated hat function.

$$\wedge(x) = \begin{cases} 10x + 1, & \text{if } -0.1 \leq x < 0 \\ -10x + 1, & \text{if } 0 \leq x < 0.1 \\ 0, & \text{elsewhere} \end{cases} \quad (3.33)$$

The differentiated and integrated functions can easily be found by applying the corresponding rules to the coefficients. After differentiation, the degree will decrease by 1 whereas the number of partitions remain the same. After integration, the degree and number of partitions will increase by 1. To continue the example from equation 3.33, figure 3.7 demonstrates the integration procedure.

As well as the hat function, the other important function to be considered is

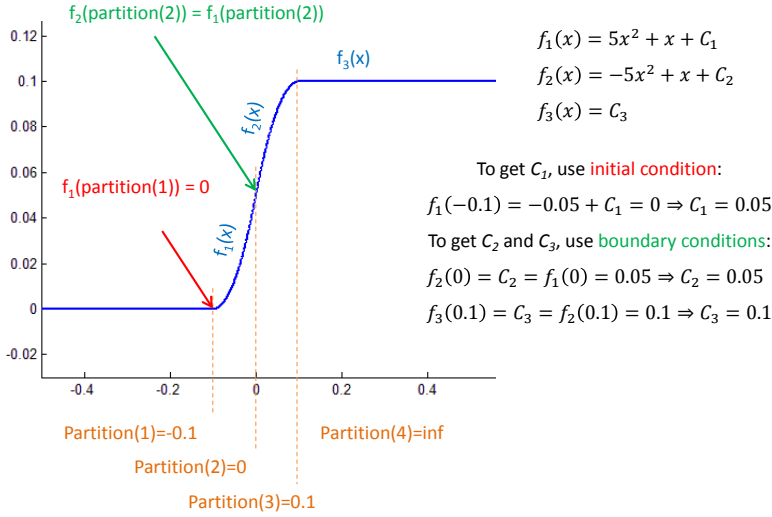


Figure 3.7: A demonstration of the integration procedure on a hat function.

the rectangular pulse, which is defined as

$$\square(x) = \begin{cases} 1, & \text{if } x \in \alpha \\ 0, & \text{elsewhere} \end{cases} \quad \text{for } \alpha = 1, 2, \dots, \Upsilon \quad (3.34)$$

Both rectangular and hat functions can be more generally described in a piecewise definition as shown in equation 3.31.

3.7.3 Spatial basis functions

Piecewise linear spatial functions must be used for the test and basis derivatives in the N_h operator, hence hat functions are used. The hat functions can be visualized at both source and observation vertices as shown in figure 3.8 where S_m represents the function at the observation/test edges, and S_n represents the function at the source/basis edges. Each sub-function index is denoted by α and β for test and basis function, respectively. For all other basis functions, rectangular pulses are used.

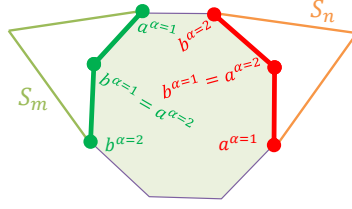


Figure 3.8: Hat function.

3.7.4 Discretised Equations

For clarity, we will explicitly show the discretised form of the BEM formulas in equation 2.48 as:

$$\begin{aligned} \square \left\{ \begin{pmatrix} \mathbf{e}_z \\ \mathbf{h}_t \end{pmatrix} \right\} &= \begin{pmatrix} \frac{\mathbf{G}}{2} + \mathbf{D} & -\frac{\eta}{c} \mathbf{S} \\ -\frac{c}{\eta} \mathbf{N} & \frac{\mathbf{G}^\top}{2} - \mathbf{D}' \end{pmatrix} \begin{pmatrix} \mathbf{e}_z \\ \mathbf{h}_t \end{pmatrix} \} \wedge + \begin{pmatrix} \mathbf{e}_z^i \\ \mathbf{h}_t^i \end{pmatrix} \} \square \\ \wedge \left\{ \begin{pmatrix} \mathbf{e}_z \\ \mathbf{h}_t \end{pmatrix} \right\} &= \begin{pmatrix} \frac{\mathbf{G}}{2} + \mathbf{D} & -\frac{\eta}{c} \mathbf{S} \\ -\frac{c}{\eta} \mathbf{N} & \frac{\mathbf{G}^\top}{2} - \mathbf{D}' \end{pmatrix} \begin{pmatrix} \mathbf{e}_z \\ \mathbf{h}_t \end{pmatrix} \} \square + \begin{pmatrix} \mathbf{e}_z^i \\ \mathbf{h}_t^i \end{pmatrix} \} \wedge \end{aligned} \quad (3.35)$$

where superscript \top denotes the transpose, and where the operators are now defined as

$$\begin{aligned} \mathbf{D}_{m,n,k} &= \int_{\Gamma} \int_{\Gamma} \square_m(\mathbf{r}) \wedge_n(\mathbf{r}') \overbrace{\left[\frac{\partial g}{\partial n'}(R, t) * T(k\Delta t - t) \right]}^{F(R,k)} dr' dr \\ \mathbf{D}'_{m,n,k} &= \int_{\Gamma} \int_{\Gamma} \wedge_m(\mathbf{r}) \square_n(\mathbf{r}') \left[\frac{\partial g}{\partial n}(R, t) * T(k\Delta t - t) \right] dr' dr \\ \mathbf{S}_{m,n,k} &= \int_{\Gamma} \int_{\Gamma} \square_m(\mathbf{r}) \square_n(\mathbf{r}') \left[g(R, t) * \frac{\partial}{\partial t} T(k\Delta t - t) \right] dr' dr \\ \mathbf{N}_{m,n,k} &= \mathbf{N}\mathbf{s}_{m,n,k} + \mathbf{N}\mathbf{h}_{m,n,k} \\ \mathbf{N}\mathbf{s}_{m,n,k} &= \frac{1}{c^2} (\hat{\mathbf{t}} \cdot \hat{\mathbf{t}}') \int_{\Gamma} \int_{\Gamma} \wedge_m(\mathbf{r}) \wedge_n(\mathbf{r}') \left[g(R, t) * \frac{\partial}{\partial t} T(k\Delta t - t) \right] dr' dr \\ \mathbf{N}\mathbf{h}_{m,n,k} &= \int_{\Gamma} \int_{\Gamma} \nabla \cdot \wedge_m(\mathbf{r}) \nabla \cdot \wedge_n(\mathbf{r}') \left[\nabla g(R, t) * \int_t T(k\Delta t - t) dt \right] dr' dr \\ \mathbf{G}_{m,n,k} &= \delta_k^0 \int_{\Gamma} \square_m(\mathbf{r}) \wedge_m(\mathbf{r}) dr \end{aligned}$$

where \sqcap and \wedge denote the rectangular and hat test/basis functions respectively, k is the current timestep, and the Kronecker delta is defined as

$$\delta_j^i = \begin{cases} 0 & (i \neq j) \\ 1 & (i = j) \end{cases}.$$

The field contribution between a pair of basis and test functions is due to interactions between all partitions. For example, testing with hat functions and using square basis functions, denoted $\langle \wedge, \sqcap \rangle$, will have 2 edge interactions ($\alpha = 1, 2$ and $\beta = 1$), whereas using hat functions for both test and basis, $\langle \wedge, \wedge \rangle$, will have 4 edge interactions. When dual basis functions are used (explained later on), the mesh has to be refined, meaning we get double the amount of edges to work with. Thus for the combination of dual hat basis and dual hat test functions, there will be 16 edge contributions.

3.7.5 Gaussian coefficient table

We can limit the spatial integration acting on the basis functions in the operators over just the interacting edges. Each edge is parameterized using $s \in [0, 1]$ so that the integral limits become 0 and 1 as shown in 3.36, where a and b are the start and end vertices for each edge, respectively.

$$\mathbf{r}(s) = a + (b - a)s \tag{3.36}$$

When finding the inner and outer integral of a parameterized function with 2 arguments, $f(x_m, x_n)$, we use Gaussian quadrature,

$$\int_{a_m}^{b_m} \int_{a_n}^{b_n} f(x_m, x_n) dx_n dx_m = l_m l_n \sum_{q_m}^{Q_m} \sum_{q_n}^{Q_n} w_{q_m} w_{q_n} f(x_m(s_{q_m}), x_n(s_{q_n})) \tag{3.37}$$

where subscripts m and n denote the outer and inner function respectively, l is the length of the discrete function, Q is the number of quadrature points to use, and w is the weight across Gaussian point q .

If we consider each basis function as a piecewise polynomial, it will consist of a subset of coefficients at different degrees, thus can be evaluated as in equation 3.31. So we can now compute the integral of the sampled function using:

$$\int_{a_m}^{b_m} \int_{a_n}^{b_n} f(x_m, x_n) dx_n dx_m = \sum_{\alpha} \Upsilon_m \sum_{\beta} \Upsilon_n l_{\alpha} l_{\beta} \sum_{p_m=0}^{P_m} \sum_{p_n=0}^{P_n} d_{\alpha, p_m} d_{\beta, p_n} \sum_{q_m=1}^{Q_m} \sum_{q_n=1}^{Q_n} \underbrace{w_{q_m} s_{q_m}^{P_m - p_m} w_{q_n} s_{q_n}^{P_n - p_n}}_{\substack{G_{\text{coeffs}_m} \quad G_{\text{coeffs}_n} \\ G_{\text{coeffs}}}} f(\mathbf{r}_m(s_{q_m}), \mathbf{r}_n(s_{q_n})) \quad (3.38)$$

where l_{α} is the length, and $d_{\alpha, p}$ is the p th coefficient of sub-function α . For computing the convolutions, we are only interested in the distances between Gaussian quadrature points. Therefore, G_{coeffs} represents a table of values which gives every possible combination of Gaussian quadrature distances, split into every combination of polynomial degree interaction.

As an example, if we use an outer linear function ($P_m = 1$) and an inner quadratic function ($P_n = 2$), along with 2 outer and 4 inner quadrature points

($Q_m = 2, Q_n = 4$), we obtain

$$\begin{aligned}
 G_{\text{coeffs}_m} &= \begin{bmatrix} w_1 s_1 & w_1 \\ w_2 s_2 & w_2 \end{bmatrix} = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \\
 G_{\text{coeffs}_n} &= \begin{bmatrix} w_1 s_1^2 & w_1 s_1 & w_1 \\ w_2 s_2^2 & w_2 s_2 & w_2 \\ w_3 s_3^2 & w_3 s_3 & w_3 \\ w_4 s_4^2 & w_4 s_4 & w_4 \end{bmatrix} = \begin{bmatrix} e & i & m \\ f & j & n \\ g & k & o \\ h & l & p \end{bmatrix} \\
 G_{\text{coeffs}} &= \begin{bmatrix} \begin{bmatrix} ae & af & ag & ah \\ be & bf & bg & bh \\ ce & cf & cg & ch \\ de & df & dg & dh \end{bmatrix} & \begin{bmatrix} ai & aj & ak & al \\ bi & bj & bk & bl \\ ci & cj & ck & cl \\ di & dj & dk & dl \end{bmatrix} & \begin{bmatrix} am & an & ao & ap \\ bm & bn & bo & bp \\ cm & cn & co & cp \\ dm & dn & do & dp \end{bmatrix} \end{bmatrix}
 \end{aligned}$$

Using Gaussian quadrature to compute the integral, a single element of the Z matrix from 3.29 becomes

$$\begin{aligned}
 Z_{m,n,k} &= \sum_{\alpha=1}^{\Upsilon_m} \sum_{\beta=1}^{\Upsilon_n} \sum_{q_m=1}^{Q_m} \sum_{q_n=1}^{Q_n} l_m^\alpha l_n^\beta w_{q_m} w_{q_n} [S_m(\mathbf{r}_m^\alpha(s_{q_m})) \cdot F(R, k) S_n(\mathbf{r}_n^\beta(s_{q_n}))] \\
 \nabla \cdot S_m \nabla \cdot S_n &= (\hat{\mathbf{t}}_m^\alpha \cdot \hat{\mathbf{t}}_n^\beta) S_m(\mathbf{r}_m^\alpha(s_{q_m})) S_n(\mathbf{r}_n^\beta(s_{q_n}))
 \end{aligned} \tag{3.39}$$

where l_n^α is the length and $\hat{\mathbf{t}}_n^\alpha$ is the tangential unit vector of the n th edge at sub-function α , w_{q_m} is the weight across Gaussian point q_m , and distance $R = |\mathbf{r}_m^\alpha(s_{q_m}) - \mathbf{r}_n^\beta(s_{q_n})|$. Q_m and Q_n is the number of Gaussian quadrature points used for the test and basis integral respectively. To use arbitrary basis and test functions, it is useful to implement the above formula in terms of

polynomial coefficients,

$$Z_{m,n,k} = \sum_{\alpha=1}^{\Upsilon_m} \sum_{\beta=1}^{\Upsilon_n} \sum_{p_m=0}^{P_m} \sum_{p_n=0}^{P_n} d_{\alpha,p_m} d_{\beta,p_n} C_{m\alpha,n\beta,k}(p_m, p_n) \quad (3.40)$$

where we sum over all test and basis function polynomial coefficients, denoted $d(S_m^\alpha)$ and $d(S_n^\beta)$ respectively, multiplied by the convolution polynomials of the same degree, C . p_m and p_n denote the current test and basis degree, and P_m and P_n denote the maximum test and basis degree. The integrated convolution polynomials for test function m and basis function n are to be preprocessed for each timestep, k , using

$$C_{m,n,k}(p_m, p_n) = l_m l_n \sum_{q_m=1}^{Q_m} \sum_{q_n=1}^{Q_n} G_{\text{coeffs}}(q_m, q_n, p_m + 1, p_n + 1) F(k, R) \quad (3.41)$$

where the convolution, F , acts on all distances between the test and basis Gaussian quadrature points $R = |\mathbf{r}_m(s_{q_m}) - \mathbf{r}_n(s_{q_n})|$. The result of $C_{m,n,k}$ for a single element will yield a $(P_m + 1) \times (P_n + 1)$ matrix of coefficients for all combinations of test and basis degrees.

3.7.6 Basis function index tables

A standard procedure for organising basis functions that span potentially several distinct objects is to use some indexing scheme. For instance, if there are multiple objects, the indexing scheme should not allow a basis function to stretch between the different surfaces. This project uses basis function index tables which specify the boundary edges that each spatial function covers.

Fig. 3.9a demonstrates a simple example creating a basis function index table using hat functions, which each stretch over 2 edges. Fig. 3.9b demonstrates creating a basis function index table using dual hat basis functions, which each stretch over 4 edges.

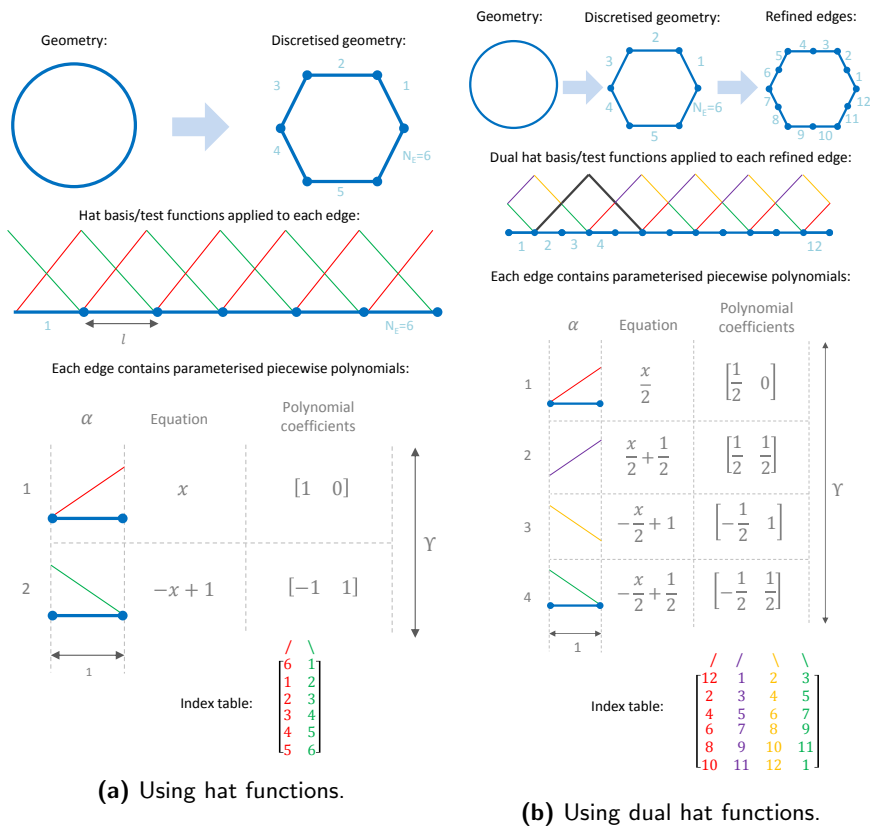


Figure 3.9: How to make the basis function index tables.

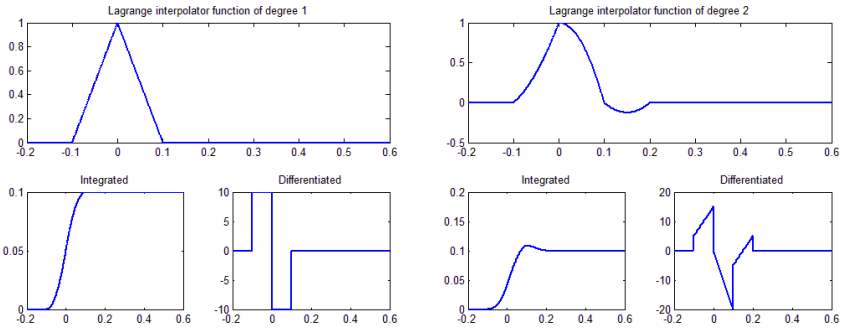


Figure 3.10: Lagrange interpolator and the associated differentiated and integrated functions, for degrees 1 and 2.

3.7.7 Temporal basis functions

The temporal basis functions are scaled versions of the Lagrange interpolator function of degree p

$$T(t) : [-1, p] \rightarrow \mathbb{R} : \frac{t}{\Delta t} \rightarrow \prod_{\phi=1}^i \frac{\phi - \frac{t}{\Delta t}}{\phi} \prod_{\phi=1}^{p-i} \frac{\phi + \frac{t}{\Delta t}}{\phi}, \quad (3.42)$$

$$if \frac{t}{\Delta t} \in [i-1, i], \quad i = 0, 1, \dots, p \quad (3.43)$$

The plots of Lagrange interpolator functions for different degrees with their associated differentiated and integrated functions are shown in Fig. 3.10.

The Lagrange interpolator function of at least degree 1 should be used for all operators. The differentiated Lagrange function will be used in operators S and N_s whereas the integrated Lagrange function will be used in operator N_h . Higher degree polynomials can be used, which will sometimes give more accurate results, but will take longer to compute and will be more susceptible to instabilities because of the dependency on a more accurate Gaussian quadrature rule to effectively sample the more complex polynomial.

Similar to the representation of the test/basis functions in 3.31, we can implement the Lagrange interpolator as a set of separate piecewise polynomi-

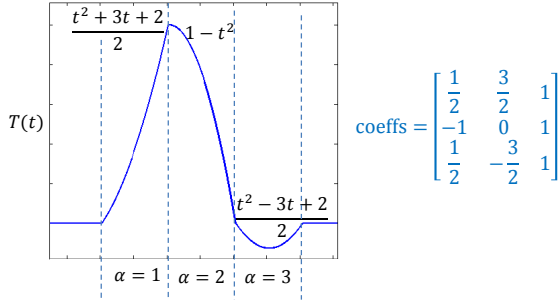


Figure 3.11: Lagrange interpolator function of degree 2 with associated quadratic equations at each interval, α , along with the matrix of coefficients.

als

$$T(t) = \begin{cases} \sum_{p=0}^P d_{\alpha,p} t^{P-p} & \text{if } t \in \alpha \\ 0, & \text{elsewhere} \end{cases} \quad (3.44)$$

where $\alpha = 1, 2, \dots, \Upsilon$.

As an example, the Lagrange interpolator of degree 2 is shown in Fig. 3.11 with its associated 2nd order quadratic equations for each partition, α , and its matrix of coefficients. The Lagrange interpolator has Υ partitions between $[-\Delta t, P\Delta t]$.

3.7.8 Temporal convolutions

Now that the construction of the piecewise polynomials for the spatial and temporal basis functions has been explained, along with the Gaussian coefficient tables, and basis function index tables, we can continue deriving the implementation from equation 3.29.

The temporal convolutions, denoted by F , are described by convolving the Green's function with the appropriate temporal basis function. The straightforward case for the 2D Green's function convolved with an arbitrary temporal basis functions is shown in 3.45.

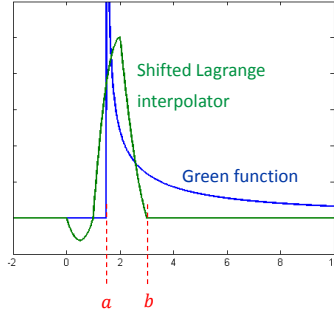


Figure 3.12: Decomposition of the temporal convolution including plots of the Green's function, and the translated Lagrange interpolator function, superimposed to demonstrate the use of the integration limits a and b .

$$F(R, k) = g(R, t) * T(t)|_{t=t_k} = \int_{R/c}^{(k+1)\Delta t} \frac{H\left(t - \frac{R}{c}\right)}{2\pi\sqrt{t^2 - \left(\frac{R}{c}\right)^2}} T(k\Delta t - t) dt \quad (3.45)$$

Equation 3.45 can be analytically solved and more generally defined for any degree:

$$F(R, k) = \frac{1}{2\pi} \sum_{\alpha=1}^{\Upsilon} \sum_{p=0}^P d_{\alpha, P-p} I_p \quad (3.46)$$

where the solved integrals are

$$\begin{aligned} I_0 &= \left[\log \left(t + \sqrt{t^2 - \left(\frac{R}{c}\right)^2} \right) \right]_a^b \\ I_1 &= \left[\sqrt{t^2 - \left(\frac{R}{c}\right)^2} \right]_a^b \\ I_p &= \frac{1}{p} \left[t^{p-1} I_1 + (p-1) \left(\frac{R}{c}\right)^2 I_{p-2} \right]_a^b \end{aligned} \quad (3.47)$$

where the updated limits, a and b become

$$\begin{aligned} a &= \max \left((k - \alpha)\Delta t, \frac{R}{c} \right) \\ b &= \max \left((k - \alpha + 1)\Delta t, \frac{R}{c} \right) \end{aligned} \quad (3.48)$$

Figure 3.12 shows the Green's function superimposed with the translated Lagrange interpolator which demonstrates the use of the integration limits, a and b . As can be seen, $g(R, t)$ is a polynomial between $[R/c, \infty]$ which is due to the Heaviside function, H . The Green's function singularity that occurs at $R = 0$ can also be seen. The translated Lagrange interpolator, $T(k\Delta t - t)$, is a polynomial between $[(k - \alpha)\Delta t, (k - \alpha + 1)\Delta t]$ for $\alpha = 1, 2, \dots, \Upsilon$. Hence the convolution will have contributions between $[a, b]$ which become the limits of integration as described in equation 3.48. The limit, a , effectively replaces the Heaviside function.

The temporally differentiated version of 3.46 is required for operators S and N_s , and the temporally integrated version is required for operator N_h . These variations can easily be found by applying the corresponding differentiation and integration rules to the coefficients. The spatial derivatives of the convolutions are required for the D and D' operators. It is important to note that the limits, as well as the integrand, in the convolution have spatial dependency. The resultant differentiated equations are shown in 3.49, 3.50, and 3.51. Because of the linear relation of the limits, the differentiated limits become Heaviside functions. For the convolution derivatives to be found, the convolution itself must be computed at the same time, which is useful to note for implementation.

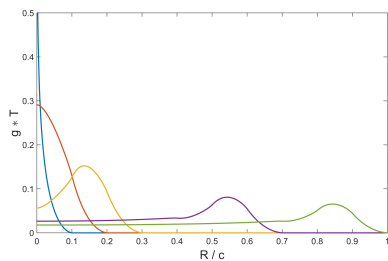
$$\partial_R F(R, k) = \frac{1}{2\pi} \sum_{\alpha=1}^{\Upsilon} \sum_{p=0}^P d_{\alpha, P-p} \partial_R I_p \quad (3.49)$$

$$\begin{aligned}
 \partial_R I_0 &= \left[\frac{\partial_R t + \frac{t}{\sqrt{t^2 - \left(\frac{R}{c}\right)^2}}}{t + \sqrt{t^2 - \left(\frac{R}{c}\right)^2}} \right]_a^b \\
 \partial_R I_1 &= \left[\frac{t}{\sqrt{t^2 - \left(\frac{R}{c}\right)^2}} \right]_a^b \\
 \partial_R I_p &= \frac{1}{p} \left[\partial_R t^{p-1} I_1 + t^{p-1} \partial_R I_1 + (p-1) \left(2 \frac{R}{c} I_{p-2} + \left(\frac{R}{c} \right)^2 \partial_R I_{p-2} \right) \right]_a^b
 \end{aligned} \tag{3.50}$$

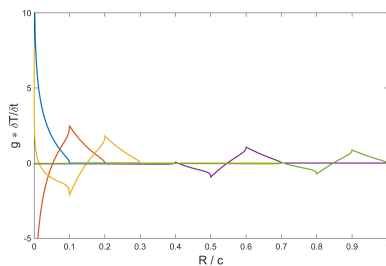
$$\begin{aligned}
 \partial_R a &= H \left(\frac{R}{c} - (k - \alpha) \Delta t \right) \\
 \partial_R b &= H \left(\frac{R}{c} - (k - \alpha + 1) \Delta t \right)
 \end{aligned} \tag{3.51}$$

Because collocation in time is used, which means the temporal testing functions are simple pulses, the temporal testing functions do not need taking into account in the implementation and will not be discussed further. More information on using more elaborate temporal testing functions can be found in [3.3].

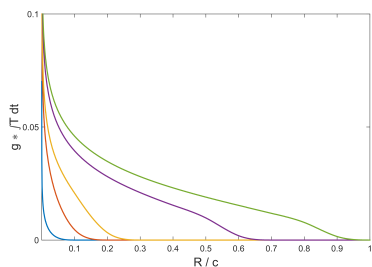
Some sample convolutions and their corresponding differentiated and integrated functions are shown in figs. 3.13a–3.13d. As can be seen, there are singularities associated with all convolutions at $k = 0$ and $R = 0$. There are additional singularities associated with the temporal differential and integral convolutions at different values of k as well. This means that the inner spatial integrals cannot be computed using Gaussian quadrature and will need to be solved analytically, or split at the singularity regions. A straightforward method of bypassing this problem is to have a different number of Gaussian quadrature points for inner and outer integrals (i.e. $Q_m \neq Q_n$) so that the points never overlap, thus R never becomes 0.



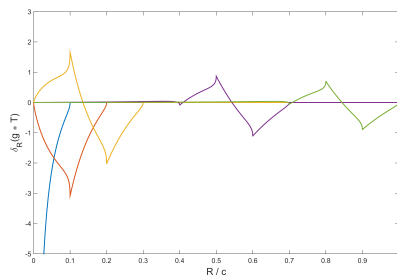
(a) No modification.



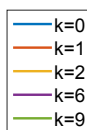
(b) Temporally differentiated.



(c) Temporally integrated.



(d) Spatially differentiated.



(e) Legend.

Figure 3.13: The Green's function convolved with the Lagrange interpolator of degree 2 at different timesteps, k .

3.8 Validation

The TD BEM code was implemented using the equations described above and applied to a cylinder meshed using 69 boundary edges was excited with a Gaussian pulse which had a maximum frequency response of ω_{max} .

The equations were evaluated at equidistant timesteps according to the following constraint

$$\Delta t = \frac{\pi}{\Phi \omega_{max}} \quad (3.52)$$

where Φ is an oversampling factor and was chosen to be 7, and the maximum frequency in radians is $\omega_{max} = 2\pi c/\lambda_{min}$.

The maximum frequency is calculated by allowing at least 10 basis functions per wavelength [3.4]. This is given by equation 3.53, where the surface is assumed to be closed, and l_{min} is the minimum edge length encountered in the discretised boundary.

$$\omega_{max} = \frac{2\pi c}{10l_{min}} \quad (3.53)$$

The number of inner and outer Gaussian quadrature points was chosen to be 5 and 4 respectively. The number of Gaussian points used when computing the integral of the incident field, $\{V_m\}$, was chosen to be 3.

The temporal basis function used in all cases was the Lagrange interpolator function of degree 1, which is the same as a hat function.

3.8.1 Scattering by a Perfect Electric Conductor

Using the equations in the table 3.1, we excite a 2D PEC cylinder with a plane wave from the left. After monitoring the electric field at the surface at

the exposed side (boundary edge nearest to the excitation source) and shadow side (boundary edge furthest from the excitation source) for a sufficient time, we compute the Fourier transform. There is an analytical solution to this problem in the frequency domain; the derivation can be found, for example, in [3.1,3.5].

The tangential current densities, J_{xy} and J_z , that were computed numerically using BEM are compared with the analytic solutions for TE and TM modes, respectively. The analytical solution for the current density can be found using 3.54 and 3.55 for the TE and TM modes respectively, where $H_n^{(2)}$ and $H_n^{(2)'} is the Hankel function of the second kind and its derivative. For computational purposes, the limits, $[-n, n]$, become $[-(ka + 10), (ka + 10)]$ to allow the end orders to sufficiently decay before the limits are reached. Further reading for derivation can be found in [3.6].$

The Hankel function differentiated with respect to the argument (ka) can be computed using equation 3.56. [3.7]

$$j_{xy}(\varphi) = \frac{2i}{\pi ka} \sum_{n=-\infty}^{\infty} \frac{i^{-n}}{H_n^{(2)'}(ka)} e^{in\varphi} \quad (3.54)$$

$$j_z(\varphi) = \frac{-2}{\pi \eta ka} \sum_{n=-\infty}^{\infty} \frac{i^{-n}}{H_n^{(2)}(ka)} e^{in\varphi} \quad (3.55)$$

$$H_n^{(2)'}(ka) = \frac{1}{2} \left(H_{n-1}^{(2)}(ka) - H_{n+1}^{(2)}(ka) \right) \quad (3.56)$$

An electric field in 2D using cylindrical coordinates has 3 components as illustrated in figure 3.14, which includes the z-directed component, E_z , the azimuthal component, E_φ , and the radial component, E_ρ . The cylinder has radius a .

When computing the time domain operators in 2.50, it is more efficient to compute them together since the convolutions and their derivatives can be calculated at the same time. Because the TE EFIE and TM MFIE require at least piecewise polynomial basis and testing functions for accurate results, all basis and testing functions are set to hat functions.

Figures 3.16a, 3.16b, 3.16c, and 3.16d show the comparison between numerical

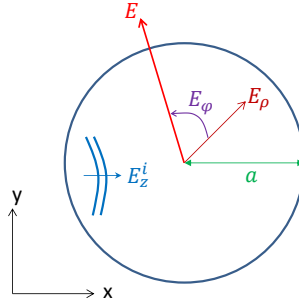


Figure 3.14: Diagram of a 2D cylinder with fields in a cylindrical coordinate system.

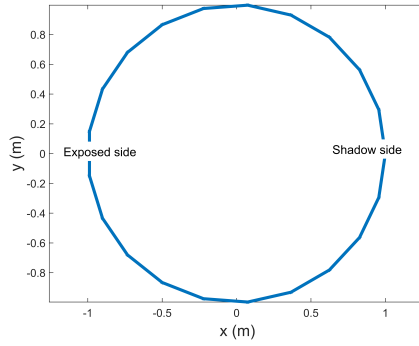


Figure 3.15: The modelled 2D cylinder showing the exposed and shadow side observation locations.

and analytical solutions for the TE EFIE, TE MFIE, TM EFIE, and TM MFIE respectively, where the shadow and exposed sides indicate the observation points demonstrated in figure 3.15. As can be seen, the error is negligible at low frequencies and only starts to deviate at the higher end of the spectrum, which is expected since this is where the Gaussian pulse frequency response will be approaching zero.

3.8.2 Scattering by a Penetrable Cylinder

The 2D cylinder is now filled with a dielectric (with $\varepsilon_r = 2$) and is excited with a plane wave. After monitoring the electric field at the surface at the exposed side and shadow side and computing the Fourier transform, we can compare the results with the analytical solution shown in equation 3.58 for the

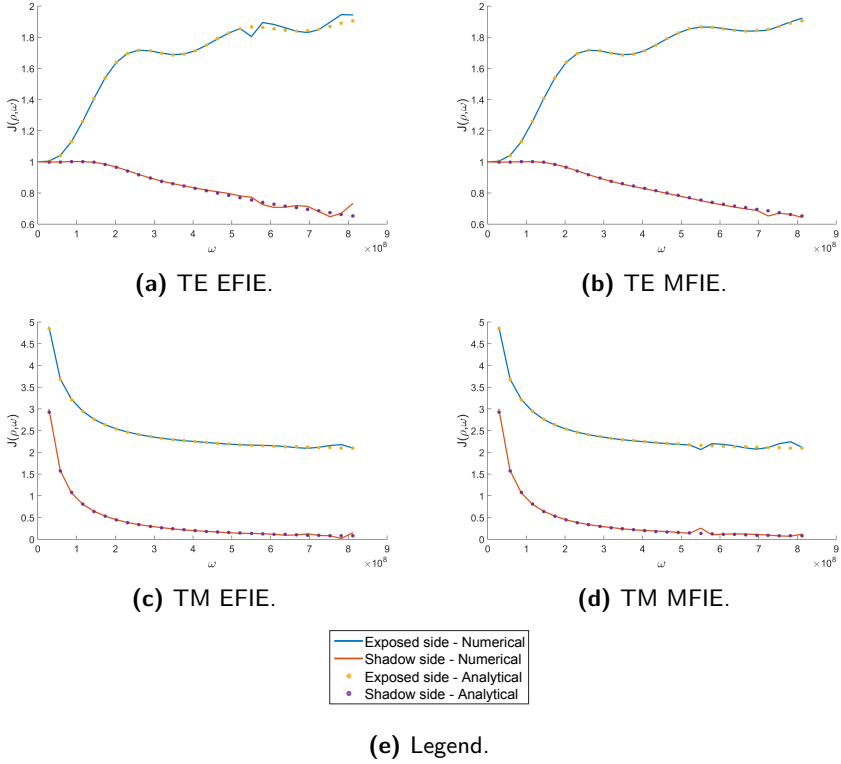


Figure 3.16: Comparison of the current densities of the exposed and shadow sides of a PEC cylinder and the corresponding analytical results.

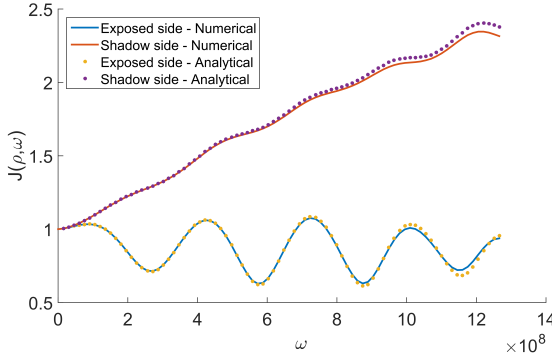


Figure 3.17: Comparison of the current densities of the exposed and shadow sides of a dielectric cylinder and the corresponding analytical results for the TM case.

TM case. J_n and J'_n is the Bessel function and its derivative, and k_{int} denotes the wave number inside the cylinder. The full derivation of these equations can be found, for example, in [3.1,3.5].

$$b_n = -i^{-n} \frac{\sqrt{\epsilon_r} J'_n(ka) J_n(k_{int}a) - \sqrt{\mu_r} J_n(ka) J'_n(k_{int}a)}{\sqrt{\epsilon_r} H_n^{(2)'}(ka) J_n(k_{int}a) - \sqrt{\mu_r} H_n^{(2)}(ka) J'_n(k_{int}a)} \quad (3.57)$$

$$j_z(\varphi) = \sum_{n=-\infty}^{\infty} b_n H_n^{(2)}(ka) e^{in\varphi} \quad (3.58)$$

Because the N operator in equation 2.48 requires at least piecewise polynomial basis functions, we will use a mixture of hat and square basis and testing functions for this case scenario.

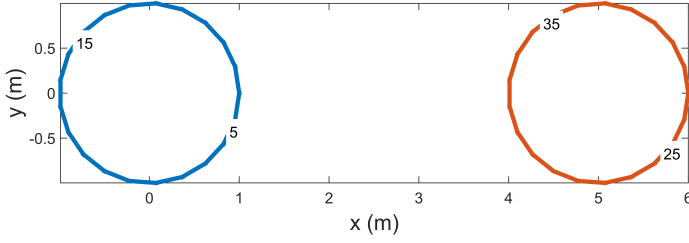
The results in figure 3.17 show good comparison between the numerical and analytical solutions for the TM case. Again, the error is negligible at low frequencies but starts to diverge at the higher end of the spectrum which is expected.

3.8.3 Scattering by 2 Penetrable Cylinders

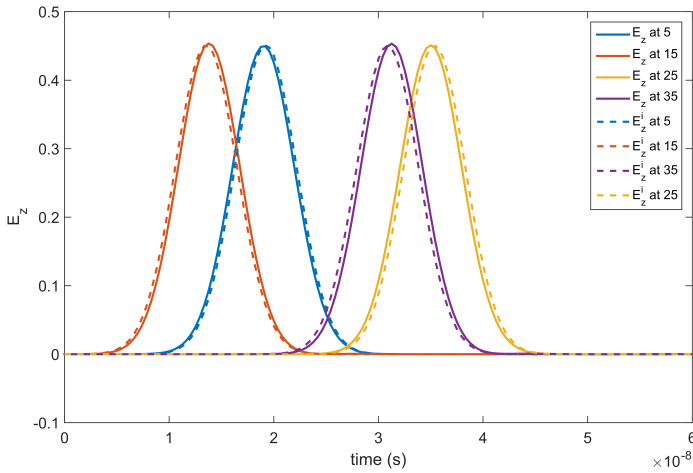
Two spatially distinct cylinders of free space are suspended in free space as shown in 3.18a. Using the implementation of equation 3.20, a plane wave

excitation from the left is observed propagating through both cylinders at the correct times as shown in 3.18b. We expect this result because no scattering should occur, thus the total field should equal the incident field.

A slight systematic displacement can be seen which is due to the choice of basis functions, in this case, a mix of hats and rectangles. Because the hat functions act at the edge centers, and the rectangle functions act at the edge vertices, the observation points for the incident wave and current density are offset by half an edge length. Normally this would not be a problem. Using dual basis functions would alleviate the inconsistency by aligning the degrees of freedom for all basis functions to the edge centers.



(a) The modelled 2D cylinders showing the locations of the observed halfedges.



(b) Comparison of the total electric field and the incident electric field.

Figure 3.18: Simulation of two spatially distinct cylinders in free space.

References

- [3.1] J.-M. J. Jin, *Theory and Computation of Electromagnetic Fields*, I. PRESS, Ed. John Wiley & Sons, 2011.
- [3.2] G. C. Hsiao and W. Wendland, “Boundary Integral Equations,” in *Theor. Numer. Anal.* Berlin: Springer, 2008.
- [3.3] Y. Beghein, K. Cools, H. Bagci, and D. De Zutter, “A Space-Time Mixed Galerkin Marching-on-in-Time Scheme for the Time-Domain Combined Field Integral Equation,” *IEEE Trans. Antennas Propag.*, vol. 61, no. 3, pp. 1228–1238, mar 2013.
- [3.4] D. Weile and G. Pisharody, “A novel scheme for the solution of the time-domain integral equations of electromagnetics,” *Antennas ...*, vol. 52, no. 1, pp. 283–295, 2004.
- [3.5] J. V. Bladel, *Electromagnetic Fields*, 2nd ed., ser. IEEE Series on Electromagnetic Wave Theory, IEEE, Ed. Wiley, 2007.
- [3.6] W. Gibson, “The Method of Moments in Electromagnetics,” nov 2007.
- [3.7] (2013, jun) Digital Library of Mathematical Functions - Bessel and Hankel Functions. Web page. [Online]. Available: <http://dlmf.nist.gov/10.6.1>

The Unstructured Transmission-Line Modelling Method

The TLM method uses lumped circuit models to temporally and spatially discretise the simulation problem into nodes. The basis of the method compares a network of transmission lines with a medium of propagating EM waves. The simplest representation of the technique uses Huygens Principle as shown in figure 4.1; each point of an advancing wave front becomes a secondary source spherical wave, which sequentially sources a new family of wave fronts and so on.

This chapter will begin by easing the reader into the TLM method by first deriving the 1D formulations. The extension to 2D unstructured TLM is not trivial and will be described in detail, starting from Maxwell's equations. Its implementation and stability will be discussed, and will include a validation test case.

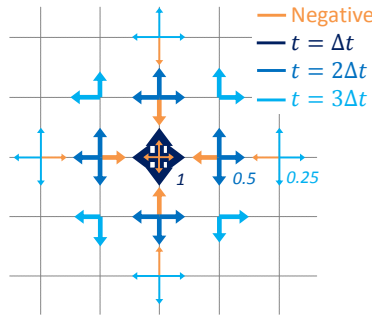


Figure 4.1: Graphical representation of Huygens' principle at different points in time, where the thickness of the lines indicates the magnitude of the wave.

4.1 1D TLM Theory

There are numerous publications that discuss the TLM method in great detail such as [4.1–4.4]. This section will focus on the 1D case.

The 1D scalar wave equation in free space (without losses), as shown in 2.16 can be compared with the equivalent circuit problem of a 1D transmission line.

Figure 4.2 shows a circuit diagram of a section from a 1D transmission line which consists of a capacitor, resistor and inductor, where L is the inductance, C is the capacitance, and R is the resistance. Assuming that the distance between sections, $\Delta x \rightarrow 0$, we can apply Kirchoff's voltage and current laws to obtain equations 4.1 and 4.2, respectively, where V is the voltage, and I is the current.

$$-\frac{\partial V}{\partial x} \Delta x = L \frac{\partial i}{\partial t} \quad (4.1)$$

$$-\frac{\partial I}{\partial x} \Delta x = C \frac{\partial v}{\partial t} + \frac{V}{R} \quad (4.2)$$

After combining these equations and eliminating I , we obtain

$$\left(\frac{\partial^2}{\partial x^2} - \frac{LC}{(\Delta x)^2} \frac{\partial^2}{\partial t^2} - \frac{L}{R(\Delta x)^2} \frac{\partial}{\partial t} \right) V = 0 \quad (4.3)$$

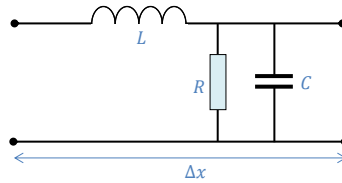


Figure 4.2: Circuit elements in an equivalent 1D transmission line.

Equations 4.1 and 4.2 can be compared with Faradays law in 1D (shown in 4.4) and the 1D TD scalar wave equation in the presence of no source currents, derived from 2.15 and shown in 4.5 where σ is conductivity.

$$-\frac{\partial e}{\partial x} = \mu \frac{\partial h}{\partial t} \quad (4.4)$$

$$\left(\frac{\partial^2}{\partial x^2} - \mu\epsilon \frac{\partial^2}{\partial t^2} - \mu\sigma \frac{\partial}{\partial t} \right) e = 0 \quad (4.5)$$

In this thesis, we are only considering low loss dielectrics, so we will ignore the diffusion-like behaviour of the 3rd term of equation 4.5 and use only the wave-like behaviour. After comparing 4.1 with 4.4, and 4.3 with 4.5, an analogy can be found. Hence we are able to obtain the equivalence between circuits and electromagnetic fields as shown in table 4.1.

The theory assumes $\Delta x \rightarrow 0$ which is an impractical resolution, but can be more realistically defined using a rule of thumb that a mesh should have a fine enough resolution to determine one tenth of the smallest wavelength that is encountered in the simulation [4.2]. This is described by equation 4.6, and is almost identical to the BEM spatial constraint described by equation 3.53.

$$\Delta x \leq \frac{\lambda_{min}}{10} \quad (4.6)$$

4.2 2D UTLM Theory

This section will derive the 2D unstructured TLM method from Maxwell's equations described in chapter 2.1.

Table 4.1: Equivalences of the field and transmission-line quantities.

Field theory			Transmission line theory			Equivalence
Quantity	Symbol	unit	Quantity	Symbol	unit	
Electric field	e	[V/m]	Voltage	V	[V]	$e \leftrightarrow -\frac{V}{\Delta x}$
Magnetic field	h	[A/m]	Current	I	[A]	$h \leftrightarrow -\frac{I}{\Delta x}$
Permittivity	ε	[F/m]	Capacitance	C	[F]	$\varepsilon \leftrightarrow \frac{C}{\Delta x}$
Permeability	μ	[H/m]	Inductance	L	[H]	$\mu \leftrightarrow \frac{L}{\Delta x}$

In TLM it is assumed that the electric and magnetic fields are constant over intervals of time Δt . For each cell, a transmission line circuit is constructed such that:

- i) the travel time on each of the constituent transmission lines equals Δt ,
- ii) the low frequency response equals that of an equally shaped region of free space,
- iii) in the lossless case, only passive or energy conserving elements are present.

We first define a triangle containing 3 link lines emanating from the circumcenter of the triangle to the edges of the triangle as shown in figure 4.3, where ϕ_α denotes the angle between the link lines. These link lines connect to neighbouring triangles via ports, each denoted by subscript α , and have link length Δ_α . Because the links connect the circumcenter of each triangle, the link lines are always perpendicular to the edges, thus allowing implicit continuity of tangential components of field values at the edges.

The scalar wave equation from equation 2.15 written in a 2D FD cylindrical coordinate system in TM mode, in terms of E_z , reduces to

$$\frac{\partial^2}{\partial r^2} E_z(r, \varphi) + \frac{1}{r^2} \frac{\partial^2}{\partial \varphi^2} E_z(r, \varphi) = k^2 E_z(r, \varphi) \quad (4.7)$$

where k ($= \omega \sqrt{\mu \varepsilon}$) is the wavenumber.

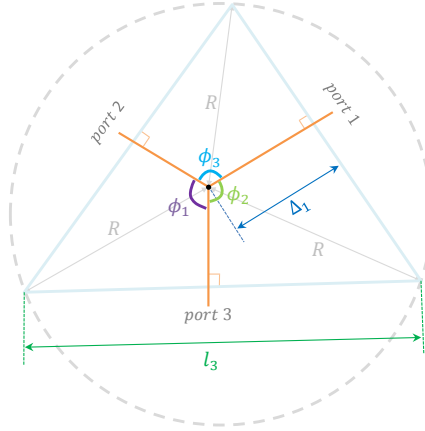


Figure 4.3: Diagram indicating lengths and angles in a UTLM triangle.

The field local to an observation point, where the mesh is sufficiently fine, can be expressed as the superposition of local solutions of the wave equation. When the mesh constraint in 4.6 is used, we can accurately approximate the solution using the first 3 modes, [4.2]

$$E_z = J_0(kr)X_1 + \cos(\varphi)J_1(kr)\frac{2X_2}{k} + \sin(\varphi)J_1(kr)\frac{2X_3}{k} \quad (4.8)$$

where J_n denotes the Bessel function of order n and X represents the expansion coefficients. This can be written in matrix form for each port as

$$\begin{pmatrix} E_{z1} \\ E_{z2} \\ E_{z3} \end{pmatrix} = \begin{pmatrix} J_0(k\Delta_1) & 2\cos(0)\frac{J_1(k\Delta_1)}{k} & 2\sin(0)\frac{J_1(k\Delta_1)}{k} \\ J_0(k\Delta_2) & 2\cos(\phi_3)\frac{J_1(k\Delta_2)}{k} & 2\sin(\phi_3)\frac{J_1(k\Delta_2)}{k} \\ J_0(k\Delta_3) & 2\cos(\phi_1 + \phi_3)\frac{J_1(k\Delta_3)}{k} & 2\sin(\phi_1 + \phi_3)\frac{J_1(k\Delta_3)}{k} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \quad (4.9)$$

We can assume the arguments of the Bessel functions are small because of the

mesh constraint in 4.6, thus can be approximated as

$$\begin{aligned}
 J_0(kr)|_{kr \ll 1} &= 1 \\
 J_1(kr)|_{kr \ll 1} &= \frac{kr}{2} \\
 \partial_r J_0(kr)|_{kr \ll 1} &= \frac{k^2 r}{2} \\
 \partial_r J_1(kr)|_{kr \ll 1} &= \frac{k}{2}
 \end{aligned} \tag{4.10}$$

Using these approximations, equation 4.9 can be reduced to

$$\overline{E_z} = \mathbf{T}_E \overline{X} \tag{4.11}$$

where $\overline{E_z}$ represents a vector of the electric field at all 3 ports, \overline{X} represents a vector of all 3 expansion coefficients, and

$$\mathbf{T}_E = \begin{pmatrix} 1 & \Delta_1 & 0 \\ 1 & \cos(\phi_3)\Delta_2 & \sin(\phi_3)\Delta_2 \\ 1 & \cos(\phi_1 + \phi_3)\Delta_3 & \sin(\phi_1 + \phi_3)\Delta_3 \end{pmatrix} \tag{4.12}$$

The modal components for the magnetic field, H_φ , can be expressed similarly using 4.4 in 2D cylindrical coordinates

$$\frac{\partial}{\partial r} E_z = -i\omega\mu H_\varphi \tag{4.13}$$

and plugging this into 4.11, along with the Bessel function derivative approximations shown in 4.10, to obtain

$$-i\omega\mu\overline{H_\varphi} = \mathbf{T}_M \overline{X} \tag{4.14}$$

where \overline{H}_φ represents a vector of the magnetic field at all 3 ports, and

$$\mathbf{T}_M = \begin{pmatrix} \frac{-k^2 \Delta_1}{2} & 1 & 0 \\ \frac{-k^2 \Delta_2}{2} & \cos(\phi_3) & \sin(\phi_3) \\ \frac{-k^2 \Delta_3}{2} & \cos(\phi_1 + \phi_3) & \sin(\phi_1 + \phi_3) \end{pmatrix} \quad (4.15)$$

Rearranging 4.11 and inserting into 4.14 to solve for the magnetic field yields an admittance relationship

$$-i\omega\mu\overline{H}_\varphi = \mathbf{T}_M \mathbf{T}_E^{-1} \overline{E}_z \quad (4.16)$$

Using the sine rules in 4.17, this can be found to be the sum of a capacitive and inductive admittance matrix shown in 4.18 where $s_\alpha = \sin(\phi_\alpha)$ is used for brevity (where $\alpha = 1, 2, 3$).

$$\begin{aligned} \sin(a)\cos(b) + \cos(a)\sin(b) &= \sin(a+b) \\ \sin(\phi_1 + \phi_3) &= -\sin(\phi_2) \end{aligned} \quad (4.17)$$

$$\begin{aligned}
 \begin{pmatrix} H_{\varphi 1} \\ H_{\varphi 2} \\ H_{\varphi 3} \end{pmatrix} &= \frac{i\omega\varepsilon}{2} \frac{\begin{pmatrix} \Delta_1\Delta_2\Delta_3s_1 & \Delta_1^2\Delta_3s_2 & \Delta_1^2\Delta_2s_3 \\ \Delta_2^2\Delta_3s_1 & \Delta_1\Delta_2\Delta_3s_2 & \Delta_1\Delta_2^2s_3 \\ \Delta_2\Delta_3^2s_1 & \Delta_1\Delta_3^2s_2 & \Delta_1\Delta_2\Delta_3s_3 \end{pmatrix}}{\Delta_2\Delta_3s_1 + \Delta_1\Delta_3s_2 + \Delta_1\Delta_2s_3} \begin{pmatrix} E_{z1} \\ E_{z2} \\ E_{z3} \end{pmatrix} \\
 &+ \frac{1}{i\omega\mu} \frac{\begin{pmatrix} \Delta_3s_2 + \Delta_2s_3 & -\Delta_3s_2 & -\Delta_2s_3 \\ -\Delta_3s_1 & \Delta_3s_1 + \Delta_1s_3 & -\Delta_1^2s_3 \\ -\Delta_2s_1 & -\Delta_1s_2 & \Delta_2s_1 + \Delta_1s_2 \end{pmatrix}}{\Delta_2\Delta_3s_1 + \Delta_1\Delta_3s_2 + \Delta_1\Delta_2s_3} \begin{pmatrix} E_{z1} \\ E_{z2} \\ E_{z3} \end{pmatrix} \quad (4.18)
 \end{aligned}$$

The equation in 4.18 is required to be reciprocal which can be satisfied by considering low frequencies, where the inductive term will be dominant. This assumption is appropriate for cases where the mesh constraint 4.6 is adhered to. However, the capacitive term must not be neglected because the inductive term can be seen to disappear when $V_1 = V_2 = V_3$. In this case, by taking a factor of Δ_α in each row of the capacitive term in equation 4.18, the numerator can be seen to equal the denominator, thus we obtain 4.19.

$$\begin{pmatrix} H_{\varphi 1} \\ H_{\varphi 2} \\ H_{\varphi 3} \end{pmatrix} = \frac{i\omega\varepsilon}{2} \begin{pmatrix} \Delta_1 & & \\ & \Delta_2 & \\ & & \Delta_3 \end{pmatrix} \begin{pmatrix} E_{z1} \\ E_{z2} \\ E_{z3} \end{pmatrix} \quad (4.19)$$

This capacitive equation can replace the one in 4.18.

Using the circuit equivalences in table 4.1, we can determine the circuit equivalences as

$$\begin{aligned}\bar{V} &= \bar{E}_z l_z \\ \bar{I} &= \bar{H}_\varphi \bar{l}\end{aligned}\tag{4.20}$$

where \bar{V} , \bar{I} , and \bar{l} are vectors of the voltage, current, and edge lengths at all 3 ports, respectively. The z-directed length, l_z , is unitary when solving 2D problems.

The circuit equivalent of 4.18 becomes

$$\bar{I} = (\bar{Y}_C + \bar{Y}_L)\bar{V}\tag{4.21}$$

where \bar{Y}_C and \bar{Y}_L denote the capacitive and inductive components of the admittance matrix respectively, and are defined as

$$\begin{aligned}\bar{Y}_C &= \frac{i\omega\varepsilon}{2} \begin{pmatrix} l_1\Delta_1 & & \\ & l_2\Delta_2 & \\ & & l_3\Delta_3 \end{pmatrix} \\ \bar{Y}_L &= \frac{1}{i\omega\mu} \frac{\begin{pmatrix} l_1(\Delta_3s_2 + \Delta_2s_3) & -\Delta_3l_1s_2 & -\Delta_2l_1s_3 \\ -\Delta_3l_2s_1 & l_2(\Delta_3s_1 + \Delta_1s_3) & -\Delta_1l_2s_3 \\ -\Delta_2l_3s_1 & -\Delta_1l_3s_2 & l_3(\Delta_2s_1 + \Delta_1s_2) \end{pmatrix}}{\Delta_2\Delta_3s_1 + \Delta_1\Delta_3s_2 + \Delta_1\Delta_2s_3}\end{aligned}\tag{4.22}$$

By applying simple geometrical rules to figure 4.3, we find that $\sin(\phi_i) = l_\alpha/(2R)$, where R is the radius of the circumcircle. Thus the inductive com-

ponent of the admittance matrix becomes

$$\overline{Y}_L = \frac{1}{i\omega\mu} \frac{\begin{pmatrix} l_1(\Delta_3 l_2 + \Delta_2 l_3) & -\Delta_3 l_1 l_2 & -\Delta_2 l_1 l_3 \\ -\Delta_3 l_2 l_1 & l_2(\Delta_3 l_1 + \Delta_1 l_3) & -\Delta_1 l_2 l_3 \\ -\Delta_2 l_3 l_1 & -\Delta_1 l_3 l_2 & l_3(\Delta_2 l_1 + \Delta_1 l_2) \end{pmatrix}}{\Delta_2 \Delta_3 l_1 + \Delta_1 \Delta_3 l_2 + \Delta_1 \Delta_2 l_3} \quad (4.23)$$

The admittance relationship in 4.21 can be compared with the nodal admittance matrix for a three bus network, as shown in figure 4.4, where the equivalent admittance matrix is

$$Y = \begin{pmatrix} y_1 + y_{12} + y_{13} & -y_{12} & -y_{13} \\ -y_{12} & y_2 + y_{12} + y_{23} & -y_{23} \\ -y_{13} & -y_{23} & y_3 + y_{13} + y_{23} \end{pmatrix} \quad (4.24)$$

The admittance values can therefore be equated as:

$$\begin{aligned} y_\alpha &= \frac{i\omega\varepsilon}{2} l_\alpha \Delta_\alpha & \text{for } \alpha = 1, 2, 3 \\ y_{12} &= \frac{1}{i\omega\mu} \frac{l_1 l_2 \Delta_3}{\Delta_2 \Delta_3 l_1 + \Delta_1 \Delta_3 l_2 + \Delta_1 \Delta_2 l_3} \\ y_{13} &= \frac{1}{i\omega\mu} \frac{l_1 l_3 \Delta_2}{\Delta_2 \Delta_3 l_1 + \Delta_1 \Delta_3 l_2 + \Delta_1 \Delta_2 l_3} \\ y_{23} &= \frac{1}{i\omega\mu} \frac{l_2 l_3 \Delta_1}{\Delta_2 \Delta_3 l_1 + \Delta_1 \Delta_3 l_2 + \Delta_1 \Delta_2 l_3} \end{aligned}$$

We can then perform a standard delta-star transformation to the inductive admittances to obtain a circuit which has 3 branches, with each branch containing an inductor link and a capacitor drain as shown in 4.5. The capacitive component of the admittance of each branch, α , can easily be obtained as $Y_{C_\alpha} = y_\alpha$.

The associated capacitance and inductance of each branch can then be found

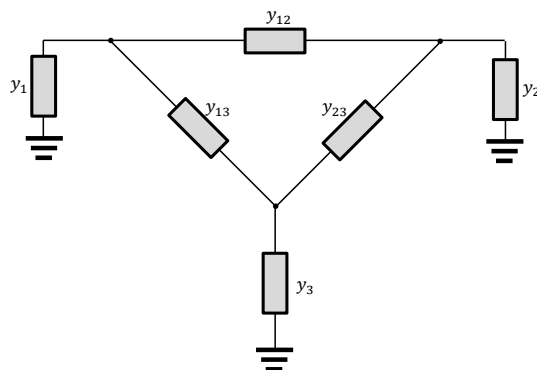


Figure 4.4: Circuit diagram for a three bus network.

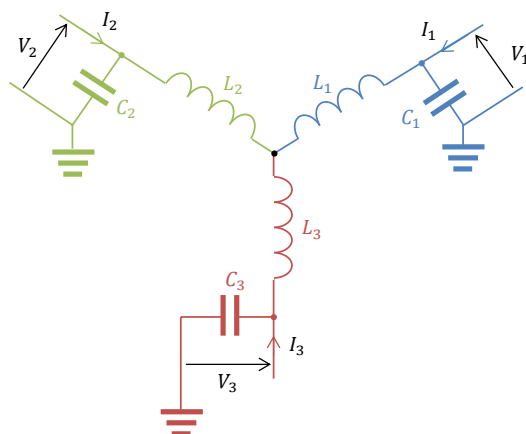


Figure 4.5: Circuit diagram for a single UTLM triangle.

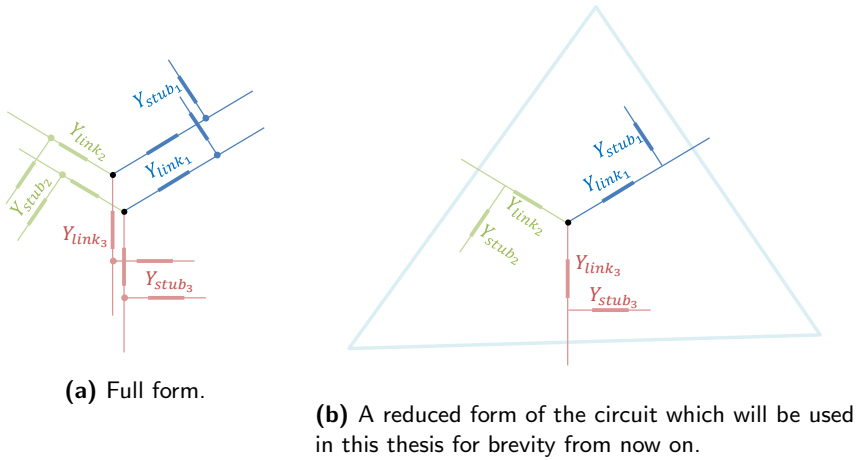


Figure 4.6: Transmission line circuit diagram for a single UTLM triangle.

using equations 4.25 and 4.26 respectively.

$$C_{\alpha} = \frac{Y_{C_{\alpha}}}{i\omega} = \frac{\varepsilon l_{\alpha} \Delta_{\alpha}}{2} \quad (4.25)$$

$$L_{\alpha} = \frac{1}{i\omega Y_{L_{\alpha}}} = \frac{\mu \Delta_{\alpha}}{l_{\alpha}} \quad (4.26)$$

We now translate the circuit in 4.5 into the transmission line circuit equivalent, which uses inductive link lines (indicated by subscript *link*) and open circuit capacitive stub lines (indicated by subscript *stub*). The resulting transmission line circuit can be seen in figure 4.6a, where figure 4.6b shows a reduced form of the circuit in a typical triangle which, for brevity, will be the form used in this thesis from now on.

To find the values of Y_{link} and Y_{stub} , we convert the circuit parameters in 4.25 and 4.26 to their time domain admittance counterparts which are shown in 4.27. Note that the transit time for a signal to traverse a link line, from circumcenter to an edge, is $\Delta t/2$.

$$\begin{aligned} Y_{L_{\alpha}} &= \frac{\Delta t/2}{L_{\alpha}} = \frac{l_{\alpha} \Delta t}{2\mu \Delta_{\alpha}} \\ Y_{C_{\alpha}} &= \frac{C_{\alpha}}{\Delta t/2} = \frac{\varepsilon l_{\alpha} \Delta_{\alpha}}{\Delta t} \end{aligned} \quad (4.27)$$

The parasitic capacitance associated with the link line inductance is

$$C_{\alpha_{error}} = \frac{(\Delta t)^2}{L_{\alpha}} \quad (4.28)$$

and can be subtracted in the stub so that the line has the correct net capacitance. This means that the stub admittance that models the line capacitance becomes 4.29, whereas the link admittance that models the line inductance is simply the inductive component of the admittance, shown in 4.30.

$$Y_{stub_{\alpha}} = Y_{C_{\alpha}} - Y_{C_{\alpha_{error}}} = \frac{\varepsilon l_{\alpha} \Delta_{\alpha}}{\Delta t} - \frac{l_{\alpha} \Delta t}{2\mu \Delta_{\alpha}} \quad (4.29)$$

$$Y_{link_{\alpha}} = Y_{L_{\alpha}} = \frac{l_{\alpha} \Delta t}{2\mu \Delta_{\alpha}} \quad (4.30)$$

To minimize dispersion error and guarantee stability, we must ensure that the stub admittance is positive, which means the timestep is constrained by

$$\Delta t < \Delta_{min} \sqrt{2\mu\epsilon}. \quad (4.31)$$

where Δ_{min} is the minimum link length used in the mesh. This constraint allows the synchronisation of signals travelling via transmission lines whose lengths that are larger than the minimum link length. In this way, every travelling signal, regardless of position in the mesh, will be located at some edge at all integer values of timestep.

4.3 Implementation

To ensure that a stable algorithm is created and a physically reasonable timestep is chosen, Delaunay triangular meshes must be used, as explained in [4.5]. This means that link lines connect each triangle circumcenter through the midpoints of each triangle edge to create a Voronoi mesh, as shown in figure 4.7a. The link lines create a 1D transmission line network which uses piecewise constant basis functions defined on the triangle edges. Unlike BEM, the basis functions are implicit in the implementation of the algorithm and will not be included here on in.

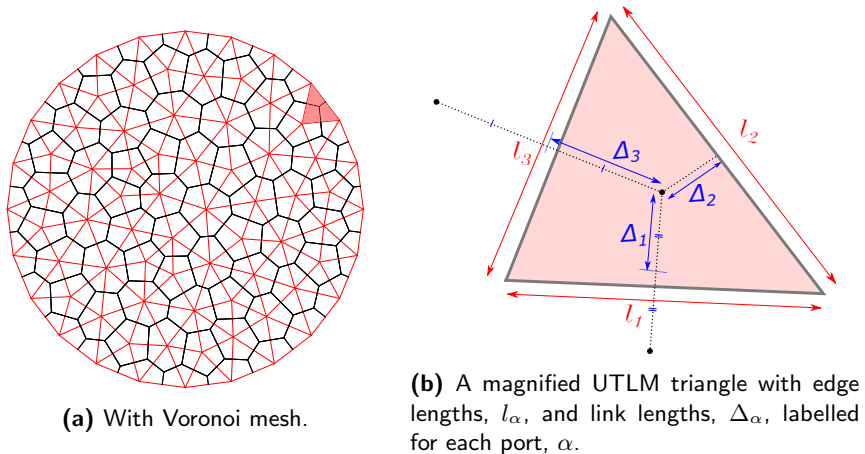


Figure 4.7: Example triangles of an unstructured mesh.

Once the transmission line network has been constructed, the response of the cell to a piecewise constant voltage signal (with respect to the time step) can be computed. The system conserves energy exactly and its solution is known analytically.

Computation of the transmission line parameters for the specified mesh using equations 4.29 and 4.30 will be performed with the dimensions indicated in figure 4.7b. Note that, to maximise the timestep and for ease of computation, the link lengths across non-boundary edges are now defined as half the distance between the connecting circumcenters.

The TLM method can be solved in many different ways, but derives its ease of use from the fact that given a piecewise constant input signal, the response can be calculated exactly using traditional transmission line techniques. Note that the transmission line description of the low frequency response of the domain automatically guarantees stability, i.e. the output energy equals the input energy.

The computation is traditionally split into two parts: the scatter process where the reflection of voltages impinging on the triangle center is computed, and the connect process where the reflection of voltages impinging on the ports is computed. Both computations are based on the construction of a Thévenin equivalent, and on the splitting of the total voltage anywhere on a line into its

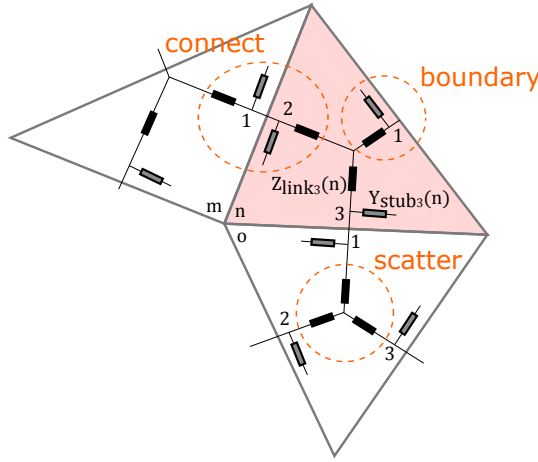


Figure 4.8: The transmission lines inside 3 connected UTLM triangles.

subsequent incident and reflected voltages,

$$V^t = V^i + V^r \quad (4.32)$$

where superscript t , i , and r denote the total, incident, and reflected values. The topology of the TLM circuits relevant to the scatter and connect steps is indicated in figure 4.8.

The implemented algorithm can be drawn as a simple flow diagram as shown in figure 4.9.

4.3.1 Scatter process

The scatter process computes the voltages reflected from the triangle center using the voltages incident from the triangle edges and from the open end of the stub lines. The voltages incident on the end of the stub lines are simply reflected, whereas the voltages reflected by the triangle center can be found using the network in figure 4.10, which makes use of equation 4.32.

$$V_{stub_\alpha}^r(n, k) = V_{stub_\alpha}^i(n, k) \quad (4.33)$$

$$V_{link_\alpha}^r(n, k) = V_0(n, k) - V_{link_\alpha}^i(n, k) \quad (4.34)$$

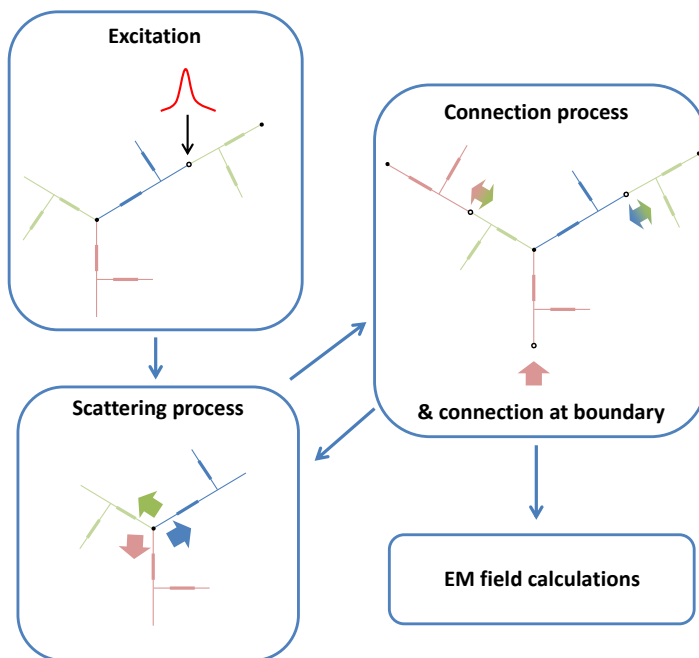


Figure 4.9: The flow diagram of the processes in a UTLM algorithm.

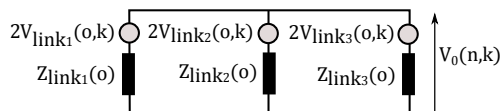


Figure 4.10: The Thevenin equivalent circuit diagram at timestep, k , for the scatter process inside triangle o .

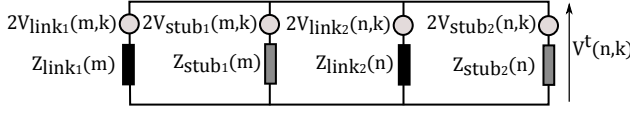


Figure 4.11: The Thévenin equivalent circuit diagram at timestep, k , for the connection between triangles m and n .

for $\alpha = 1, 2, 3$, where superscript i and r denote the incoming and reflected voltages respectively, and $V_0(n, k)$ is the total voltage in the centre of triangle n at timestep k .

Investigation of the Thévenin equivalent circuit as shown in figure 4.10 gives the total voltage at the centre of a triangle,

$$V_0(n, k) = \frac{2 \sum_{\alpha=1}^3 V_{link_{\alpha}}^i(n, k) Y_{link_{\alpha}}(n)}{\sum_{\alpha=1}^3 Y_{link_{\alpha}}(n)}. \quad (4.35)$$

4.3.2 Connect process

In the connect process, the voltages reflected at the inter-triangle ports are computed.

Using nodal analysis on the Thévenin equivalent circuit as shown in figure 4.11, and equation (4.32), the total voltage at the port between triangles m and n is

$$V^t(m, k) = V^t(n, k) = \frac{2 \left[V_{link1}^r(m, k) Y_{link1}(m) + V_{stub1}^r(m, k) Y_{stub1}(m) + V_{link2}^r(n, k) Y_{link2}(n) + V_{stub2}^r(n, k) Y_{stub2}(n) \right]}{Y_{link1}(m) + Y_{stub1}(m) + Y_{link2}(n) + Y_{stub2}(n)}.$$

The incident link and stub voltages for the next timestep are

$$V_{link_{\alpha}}^i(n, k+1) = V^t(n, k) - V_{link_{\alpha}}^r(n, k) \quad (4.36)$$

$$V_{stub_{\alpha}}^i(n, k+1) = V^t(n, k) - V_{stub_{\alpha}}^r(n, k) \quad (4.37)$$

for $\alpha = 1, 2, 3$. These values are then used in the scatter process at the next

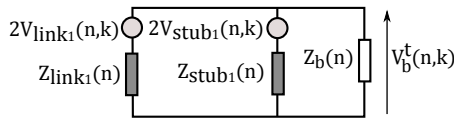


Figure 4.12: The Thevenin equivalent circuit diagram at timestep, k , for the connection at a boundary edge of triangle n .

timestep.

4.3.3 Connection at the boundaries

To model the radiating behaviour of the fields at the TLM boundary of the problem space, the simplest approach is to terminate the mesh with a lumped impedance, the so called matched impedance, with value equal to the wave impedance of free space. This is indicated in the circuit of fig. 4.8c. In this situation the total voltage and current at the exterior edge becomes

$$\begin{aligned} V_b^t(n, k) &= \frac{I_b^t(n, k)}{Y_{link_1}(n) + Y_{stub_1}(n) + Y_b} \\ I_b^t(n, k) &= 2V_{link_1}^r(n, k)Y_{link_1}(n) \\ &\quad + 2V_{stub_1}^r(n, k)Y_{stub_1}(n) \end{aligned} \quad (4.38)$$

where subscript b indicates values at the boundary edge, and Y_b is the boundary admittance.

The boundary admittance can be calculated using the reflection coefficient, Γ as shown in equation 4.39. For a matched, or radiating, boundary condition, $\Gamma = 0$. For an open circuit boundary condition, $\Gamma = 1$.

$$Y_b = Y_{fs} \frac{1 - \Gamma}{1 + \Gamma} \quad (4.39)$$

The characteristic admittance of free space, Y_{fs} , is given by

$$\begin{aligned} Y_{fs} &= \frac{I_{fs}}{V_{fs}} \\ &= \frac{l}{\eta_0} \end{aligned} \quad (4.40)$$

where V_{fs} and I_{fs} represent the equivalent voltage and current, respectively, due to a field propagating in free space, and the free space impedance is $\eta_0 = \sqrt{\mu_0/\varepsilon_0}$. This can be confirmed by finding the characteristic admittance of a transmission line in free space, which is

$$\begin{aligned} Y_{TL} &= Y_{link} + Y_{stub} \\ &= \frac{\Delta_\alpha}{\Delta t} \varepsilon_0 l_\alpha \\ &= c_0 \varepsilon_0 l_\alpha \\ &= \frac{l_\alpha}{\eta_0} \end{aligned} \tag{4.41}$$

where c_0 is the wave propagation in free space.

Alternatively, one can simulate a PEC boundary by replacing the boundary impedance in figure 4.12 with a short circuit. In this case, the updated incident voltages would simply become

$$V_{link_\alpha}^i(n, k+1) = \Gamma V_{link_\alpha}^r(n, k) \tag{4.42}$$

$$V_{stub_\alpha}^i(n, k+1) = \Gamma V_{stub_\alpha}^r(n, k) \tag{4.43}$$

for $\alpha = 1, 2, 3$.

The matched boundary condition placed at each surface edge effectively reduces the radiation problem to one dimension. Unfortunately, this is a crude approximation to a perfect radiating boundary condition that is inaccurate at non-smooth boundaries and for obliquely incident fields. Current implementations of TLM get around this shortfall by enlarging the meshed domain so that the reflections do not reach the observed area, but at significant computational cost.

There are other methods to improve the boundary conditions, such as using Perfectly Matched Layers (PMLs) as described in [4.6,4.7], but these methods currently cannot reach the level of performance that would make them comparable with Finite-Difference Time-Domain (FDTD) PML algorithms [4.8]. This is because the ideal numerical conductivity values required to construct the PML material are difficult to implement in TLM.

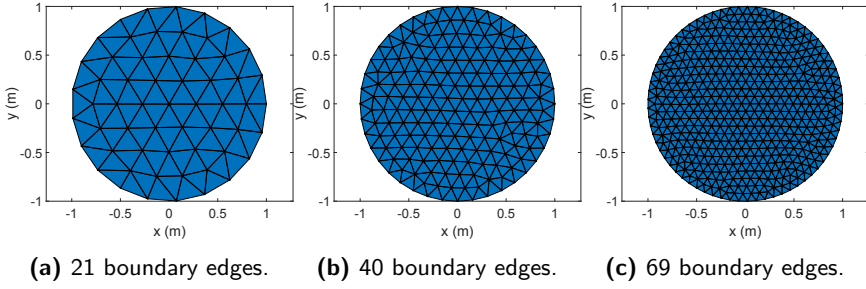


Figure 4.13: A 1m cylindrical cavity in 2D meshed using Delaunay criteria.

4.4 Validation

A test using the 2D UTLM algorithm was performed by applying a Gaussian pulse excitation to a random node in a cylindrical PEC cavity, and the results compared with the analytical resonant frequencies. The cavity is then more finely meshed and the test is repeated to monitor convergence. The cylindrical cavity meshes that were used had a radius, $a = 1$, and was discretised using 21, 40 and 69 boundary edges as shown in figure 4.13.

The Delaunay meshes were created using a Matlab library, details of which can be found in appendix A. Meshing of arbitrary structures that satisfy the Delaunay condition is not a trivial task and the details for which is outside the scope of this thesis, but there are tools available that can help; for examples, the reader is directed to appendix A.

To obtain the analytical results, the Helmholtz equation for TM fields using cylindrical coordinates as shown in 4.44 must be solved. The solution is comprised of a radial, ρ , component and an azimuthal, φ , component as shown in 4.45 where J_m is the m th order Bessel function of the first kind, and the field components are depicted in figure 4.14.

$$\left(\frac{\partial^2}{\partial \rho^2} + \frac{1}{\rho} \frac{\partial}{\partial \rho} + \frac{1}{\rho^2} \frac{\partial^2}{\partial \varphi^2} \right) E_z + k^2 E_z = 0 \quad (4.44)$$

$$E_z(\rho, \varphi) = (A \sin(m\varphi) + B \cos(m\varphi)) J_m(\rho k) \quad (4.45)$$

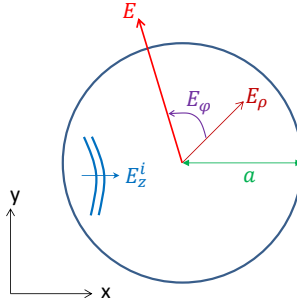


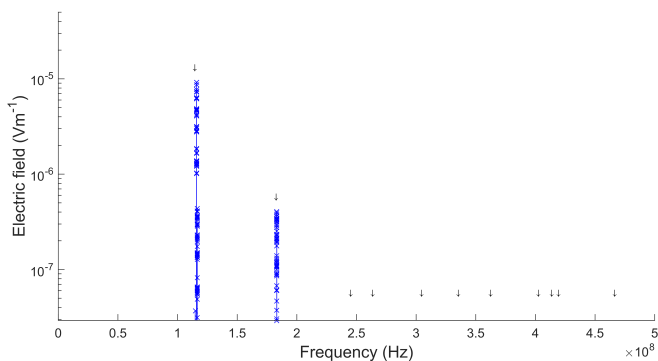
Figure 4.14: Cylindrical field components.

The resonant modes of the cavity are determined by the boundary condition, $\rho = a$, hence we must solve $J_m(ak) = 0$, which consequently gives us the resonant frequencies,

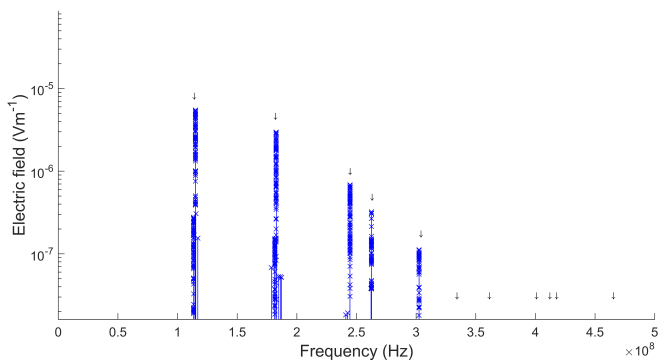
$$(f_r)_{mn} = \frac{\chi}{2\pi a \sqrt{\mu\epsilon}} \quad (4.46)$$

where χ represents the Bessel function zeros, and m and n denotes the Bessel function order and root, respectively.

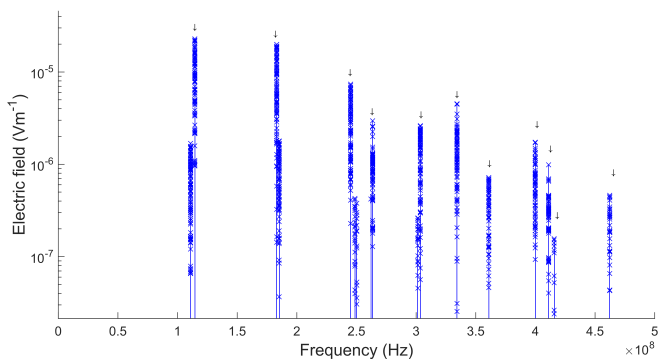
A Fourier transform was applied to the resultant fields obtained from multiple observation points in the UTLM solver to monitor convergence at specific frequencies. These frequency domain results are shown via stems in 4.15 and the analytical results (calculated using 4.46) are shown via downwards pointing arrows. The graphs in 4.15 show the results for increasingly fine meshes. Each graph indicates increasingly excellent agreement as the mesh becomes more fine.



(a) Results using mesh with 21 boundary edges.



(b) Results using mesh with 40 boundary edges.



(c) Results using mesh with 69 boundary edges.

Figure 4.15: Comparison between analytical (black arrows) and numerical (blue) resonant frequencies at different observation points for different meshes.

References

- [4.1] C. Christopoulos, *The Transmission-Line Modeling Method TLM*, ser. The IEEE Series on Electromagnetic Wave Theory. Wiley, 1995.
- [4.2] —, *The Transmission-Line Modeling (TLM) Method in Electromagnetics*. Morgan & Claypool Publishers, 2006.
- [4.3] S. Ramo, J. R. Whinnery, and T. V. Duzer, *Fields and Waves in Communication Electronics*, 2nd ed. John Wiley & Sons, 1984.
- [4.4] H. Wakatsuchi, “Computational Studies of the Electromagnetic Properties of Metamaterials and Applications,” Ph.D. dissertation, University of Nottingham, jul 2011.
- [4.5] P. Sewell, J. Wykes, T. Benson, C. Christopoulos, D. Thomas, and A. Vukovic, “Transmission-line modeling using unstructured triangular meshes,” *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 5, pp. 1490–1497, 2004.
- [4.6] J. Paul, C. Christopoulos, and D. Thomas, “Perfectly matched layer for transmission line modelling (TLM) method,” *Electron. Lett.*, vol. 33, no. 9, p. 729, 1997.
- [4.7] J. L. Dubard and D. Pompei, “Optimization of the PML Efficiency in 3-D TLM Method,” *IEEE Trans. Microw. Theory Tech.*, vol. 48, no. 7 PART 1, pp. 1081–1088, 2000.
- [4.8] N. Pena and M. Ney, “Absorbing-boundary conditions using perfectly matched-layer (PML) technique for three-dimensional TLM simulations,” *IEEE Trans. Microw. Theory Tech.*, vol. 45, no. 10, pp. 1749–1755, 1997.

Review of Previous Hybridisation Attempts

Previous TLM-BEM hybrids have been attempted. This section of the thesis will describe the methods that have previously been published in various conferences and journals. This will include their advantages and disadvantages, and overall conclusions.



5.1 The TLM-IE method

In the TLM-IE method as described by Pierantoni which can be found in references [5.1,5.2], objects are encompassed within near-field TLM subdomains as shown in figure 5.1. The tangential electromagnetic field components on the subdomain boundary represent equivalent surface currents (as described in the equivalence theorem). These equivalent sources are coupled in the external free-space region using the Green's function and solved using MoM.

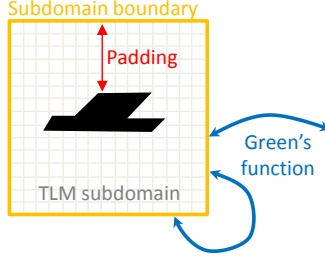


Figure 5.1: Diagram depicting the TLM-IE method.

The point source field, ${}^{TLM}\mathbf{E}_t^{inc}$, is placed in the near-field region and is calculated on the subdomain boundary using TLM. Tangential radiated fields, \mathbf{E}_t^r , in free space are calculated on the boundary using Green's functions.

By applying continuity, the total tangential fields at the subdomain boundary are found to be

$$\mathbf{E}_t = {}^{TLM}\mathbf{E}_t^{inc} + \mathbf{E}_t^r \quad (5.1)$$

The 3D TD EFIE and MFIE is used to solve for the radiated field. The electric field can be defined in compact dyadic form:

$$\begin{pmatrix} \mathbf{E}_t(\mathbf{r}, t) \\ \mathbf{H}_t(\mathbf{r}, t) \end{pmatrix} = \begin{pmatrix} C_e & C_h \\ D_e & D_h \end{pmatrix} \begin{pmatrix} \mathbf{E}_t(\mathbf{r}', t) \\ \mathbf{H}_t(\mathbf{r}', t) \end{pmatrix} + \begin{pmatrix} {}^{TLM}\mathbf{E}_t^{inc} \\ {}^{TLM}\mathbf{H}_t^{inc} \end{pmatrix} \quad (5.2)$$

where C_e , C_h , D_e and D_h represent the integral and differential operators. The radiated field includes a coefficient, T , which changes according to self-interactions; $T = 2$ when calculating the boundary self-interactions, and $T = 1$ when calculating interactions from external structures as proven in [5.3]. This is assumed to be an alternative to using the Gram matrix.

The discretised form of 5.2 uses rectangular/pulse functions for the surface basis and time basis functions, Φ and P respectively as demonstrated in equation

5.3.

$$\begin{aligned}
 E_t(r, t) &= \sum_{m'=1}^M \sum_{n'=0}^N E_\phi(r_{m'}, t_{n'}) \Phi(r - r'_m) P(t - t'_n) \\
 E_\phi(m, n) &= \sum_{m'=1}^M \sum_{n'=0}^N \left[\mathbf{C}_e(m, m', n - n') E_\phi(m', n') + \right. \\
 &\quad \left. \mathbf{C}_h(m, m', n - n') H_\psi(m', n') \right] + {}^{TLM} E_t^{inc}(m, n)
 \end{aligned} \tag{5.3}$$

where E_ϕ and H_ψ are the unknown expansion coefficients.

As can be seen in 5.3, the total field at time $t = t_n$ can be directly computed from the incident field at the same time and the past history of tangential fields in all cells from $t = 0$ to $t = t_{n-1}$.

Testing is then applied to 5.3 with Dirac delta weighting functions, where the weights are defined as

$$W_{mn}(r_m, t_n) = \delta(t - n\Delta t) \delta(r - r_m) \tag{5.4}$$

Because of the choice of basis and testing functions, a smooth field distribution is assumed on the subdomain boundary. This means that padding is required between the object and subdomain boundary. The novel method described in chapter 6 of this thesis is able to solve this problem by using expansion functions of the correct regularity via the introduction of a dual mesh.

The steps of the TLM-IE algorithm are as follows:

- 1) TLM starts with absorbing boundary conditions on the subdomain boundary.
- 2) For any timestep, t_n , we know the incident field at the subdomain boundary.
- 3) For a particular observation cell outside the TLM subdomain (m) there will be a contribution from a source cell on the boundary (n) using integral equation 5.3 after testing. This contribution will be in effect

when the current time satisfies the condition $t_n \geq R_{mn}/c$ where $R_{mn} = |r_m - r_n|$.

- 4) Repeat for every observation cell for every timestep.
- 5) For every timestep, the corrected tangential field provides new boundary conditions for the TLM algorithm on the subdomain boundary.

5.2 The Adapted Radiating Boundaries (ARB) method

In the TLM-IE method, the field components on the subdomain boundaries are expanded/approximated using rectangular basis functions. However, the approximations lead to discretisation errors on the subdomain boundaries and so a distance has to be kept (padding) between the objects and boundaries in order to ensure a smooth field distribution along the boundaries.

An extension to the TLM-IE method was found by using the discrete time-domain TLM-Green's function (or Johns matrix) to decrease the padding of free space around the scatterer and enable more accurate open boundary conditions. [5.4–5.6]

At the boundary of 2 structures, the reflected voltages of the observed “removed branches” are defined using Johns matrix [5.7]

$$\begin{pmatrix} V_1^r(m, k) \\ V_2^r(m, k) \end{pmatrix} = \begin{pmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{21} & \mathbf{G}_{22} \end{pmatrix} * \begin{pmatrix} V_1^i(m, k') \\ V_2^i(m, k') \end{pmatrix} \quad (5.5)$$

where the diagonal submatrices, \mathbf{G}_{11} and \mathbf{G}_{22} represent reflection (self-interaction) matrices, whilst \mathbf{G}_{12} and \mathbf{G}_{21} represent transmission matrices.

Figure 5.2 demonstrates the signal paths between two objects. Both objects are encompassed by a TLM subdomain Free Space Boundary (FSB) surface, S_B , and have an intermediary Equivalent Source (ES) surface, S_S that can

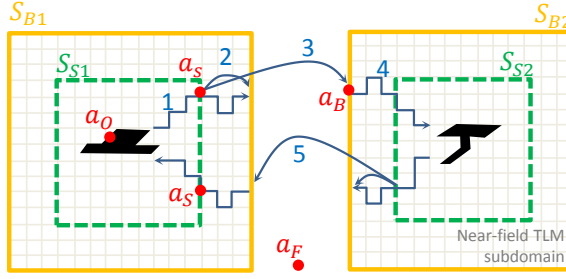


Figure 5.2: Diagram showing the signal paths and wave amplitudes between two arbitrary objects (which can now fill S_S).

now be filled completely by the objects. The minimal distance between the FSB surface and the ES surface is now 1 cell, making the technique quicker than the previously proposed TLM-IE method. The coupling between these objects occurs in multiple paths as shown in 5.2. The journeys of these paths and their computation method is as follows:

- 1) From object 1 to S_{B1} using TLM.
- 2) From S_{S1} to S_{B1} using the discrete TLM-Green's function (for the radiating boundary condition).
- 3) From S_{S1} to S_{B2} using \mathbf{G}_{21} of the dyadic Green's function.
- 4) From S_{B2} to object 2 using TLM.

This process is repeated for all objects. When computing the path from S_{S2} to S_{B1} , the \mathbf{G}_{12} interaction of the dyadic Green's function would be used. Hence, only the external contributions of the Green's functions are needed because self-interactions are computed using the TLM algorithm.

Wave amplitudes refer to the voltage values used in the TLM algorithm; $a \equiv V^i$, $b \equiv V^r$.

- a_F refers to the waves propagating within free space.
- a_O refers to the waves propagating within the object region.

- a_B refers to the waves propagating on the FSB surface (coefficients of a_F).
- a_S refers to the waves propagating on the ES surface from free space.
- a_o refers to the waves propagating on the ES surface from the object.

Using these variables, the TLM scattering and connection matrices are found as shown in equations 5.6 and 5.7, respectively.

$$\begin{pmatrix} b_F(j) \\ b_S(j) \\ b_s(j) \\ b_O(j) \end{pmatrix} = \begin{pmatrix} \mathbf{S}_F & \mathbf{S}_{FS} & - & - \\ \mathbf{S}_{SF} & \mathbf{S}_S & - & - \\ - & - & \mathbf{S}_s & \mathbf{S}_{sO} \\ - & - & \mathbf{S}_{Os} & \mathbf{S}_O \end{pmatrix} \begin{pmatrix} a_F(j-1) \\ a_S(j-1) \\ a_s(j-1) \\ a_O(j-1) \end{pmatrix} \quad (5.6)$$

$$\begin{pmatrix} a_F(j) \\ a_S(j) \\ a_s(j) \\ a_O(j) \end{pmatrix} = \begin{pmatrix} \mathbf{C}_F & - & - & - \\ - & - & \mathbf{I} & - \\ - & \mathbf{I} & - & - \\ - & - & - & \mathbf{C}_O \end{pmatrix} \begin{pmatrix} b_F(j) \\ b_S(j) \\ b_s(j) \\ b_O(j) \end{pmatrix} \quad (5.7)$$

where \mathbf{S} , \mathbf{C} and \mathbf{I} denote the scatter, connection and unit matrix, respectively.

Using equations 5.6 and 5.7, a matrix of discrete Green's functions can be derived, as shown in equation 5.8, to obtain the wave amplitudes anywhere in free space at timestep j using equation 5.9.

$$\mathbf{G}(j-i) = (\mathbf{C}_F \mathbf{S}_F)^{j-1-i} (\mathbf{C}_F \mathbf{S}_{FS}) \quad (5.8)$$

$$a_F(j) = \sum_{i=0}^{j-1} G(j-1) a_S(i) \quad (5.9)$$

All the waves in free space are dependent on the previous wave amplitudes detected on the ES surface. Using this knowledge, the wave amplitudes on

the FSB surfaces can be found using the combination of signals coming from the interior structure (along with those from free space) and the equivalent sources radiated from the exterior structures,

$$\mathbf{a}_B = \mathbf{a}_F|_{FSB} + \frac{1}{2}(-\hat{\mathbf{n}} \times \hat{\mathbf{n}} \times \mathbf{E}_B^r + Z \hat{\mathbf{n}} \times \mathbf{H}_B^r) \quad (5.10)$$

where the tangential fields radiated onto the ES surface from the exterior structure, \mathbf{E}_B^r and \mathbf{H}_B^r , are calculated using the continuous Green's functions as in the usual IE method.

5.3 Modelling thin wire structures

In reference [5.8], Lindenmeier used a technique similar to the one used in his other papers [5.1,5.2,5.4–5.6] incorporating the Green's function to find the current on a thin PEC wire structure. The Schelkunoff Huygens representation of the equivalence theorem was used, where surface currents are defined by the tangential field components on the surface and thus are sources of radiation.

A hybrid method combining TLM and MoM was used by Khelifi in [5.9] which closely resembles the TLM-IE method but is used to analyse the interaction between a thin wire and a complex object. The complex object was discretised in a TLM subregion surrounded by an imaginary boundary whilst the thin wire was some distance away. By applying continuity of the field at the TLM boundary, the electric field at the boundary can be found as

$$E_B = E_{Bw}^r + E_{Bself}^r + E_B^i \quad (5.11)$$

where E_{Bw}^r is the field radiated by the wire onto the boundary, E_{Bself}^r is the field radiated by the boundary onto itself, and E_B^i is the incident field at the boundary (a point source emanating from inside the TLM subregion). The radiated fields, E_{Bself}^r , are calculated using MoM with the EFIE and MFIE equations multiplied by a coefficient, T which is either 2 for self interactions or 1 otherwise. The field values of the self-interactions on the TLM boundary are expanded using rectangular basis functions.

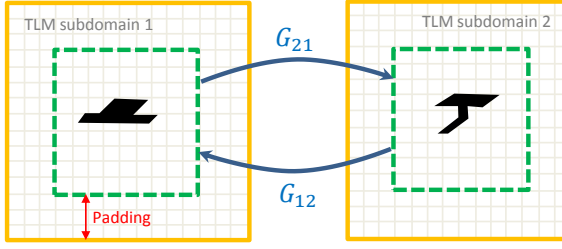


Figure 5.3: Diagram of the IRIS method using only 2 components of the Green's dyad.

To inject the waves back into the TLM subregion, equation 5.12 is used, obtained from [5.10].

$$a_m = \frac{1}{2}(-\hat{\mathbf{n}}_m \times \hat{\mathbf{n}}_m \times \mathbf{E}_{Bm}^r + Z \hat{\mathbf{n}}_m \times \mathbf{H}_{Bm}^r) \quad (5.12)$$

where the wave amplitudes, a_m , on those transmission lines cut by the TLM subregion boundary at patch m , are mapped by the field values at that point.

5.4 IRIS (Interference and Radiation of Internal Surfaces) method

The Interference and Radiation of Internal Surfaces (IRIS) method described in [5.11] is almost exactly the same as the TLM-IE method except that the internal surfaces are coupled to each other using only the G_{12} and G_{21} components of the TD dyadic Green's function.

Self-interactions are computed within the TLM algorithm. The outer surfaces/external boundaries use traditional ABCs for termination of the TLM subdomain, hence a large amount of padding is used so spurious reflections do not interfere with the internal domain as depicted in figure 5.3.

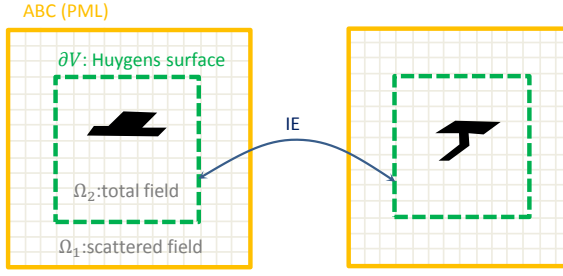


Figure 5.4: Diagram showing the different regions in the TF/SF technique.

5.5 Total-Field/Scattered-Field (TF/SF) Technique

In reference [5.12], Fichtner uses elements of the TLM-IE method to model a Huygens surface surrounding a TLM subdomain. On this surface (denoted as ∂V), there exists current distributions in accordance with the Huygens-Schellkunoff representation of the equivalence theorem. The TF/SF method separates the simulation domain into regions as shown in 5.4. The inner region, Ω_2 , contains incident and scattered fields, whereas the outer region, Ω_1 , only contains scattered fields. This is a consequence of using the Huygens surface and integral equation method. This means that arbitrary sources can be used for excitation [5.13], but no sources can be placed inside region 1.

The IE method is used between the ∂V of different structures, where the factor T again is used and equates to 2 when source and observation points lie on ∂V . The EFIE and MFIE are expanded with 2D pulse basis functions in both space and time, and point-matching is used for testing.

The fields are mapped at ∂V using a special connection process which forces the TLM wave amplitudes in region 1 to zero,

$$a = \mathbf{\Gamma}b + \frac{1}{2}(\mathbf{\Gamma}_E E_B + Z_0 \mathbf{\Gamma}_H H_B) \quad (5.13)$$

where a and b refer to the incident and reflected wave amplitudes, and $\mathbf{\Gamma}$ denotes the connection matrix which depends on the TLM node location; i.e. if it is located in the total field region or the scattered field region.

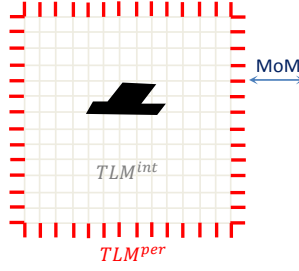


Figure 5.5: Diagram of the hybridised FD-TLM-IE approach where there are internal and perimeter TLM nodes.

5.6 Hybridisation of 2D FD-TLM with 2D-EFIE

In reference [5.14], Zedler uses four-port shunt TLM nodes (structured 2D TM propagation) in the frequency domain, where the currents at the perimeter and the interior of the TLM subdomain, as shown in figure 5.5, can be expressed as

$$\begin{pmatrix} I^{per} \\ I^{int} \end{pmatrix} = \begin{pmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} \end{pmatrix} \begin{pmatrix} V^{per} \\ V^{int} \end{pmatrix} \quad (5.14)$$

where each 4×4 admittance matrix is defined for isotropic materials as

$$\mathbf{Y}^{-1} = \frac{1}{2}i\omega\mu d \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} + \frac{\mathbf{1}}{i\omega\epsilon d} \quad (5.15)$$

where d is the TLM node length and $\mathbf{1}$ is a 4×4 matrix of ones.

Using the associated 2D EFIE equation for TM propagation of the scattered electric field in the frequency domain,

$$\begin{aligned} V^{MoM}(r) &= \frac{1}{2}V(r) - \int_{\Gamma_0} \frac{\partial \mathbf{G}(r, r')}{\partial n'} V(r') dr' + i\omega\mu \int_{\Gamma_0} \mathbf{G}(r, r') I(r') dr' \\ &= \left(\frac{\mathbf{1}}{2} - \mathbf{D} \right) V^{per} - i\omega\mu \mathbf{S} I^{per} \end{aligned} \quad (5.16)$$

where \mathbf{S} and \mathbf{D} are the Neumann and Dirichlet operators respectively, Γ_0 is the TLM subdomain boundary, and V^{MoM} is the plane wave incident field. The discretised equation uses pulse expansion basis functions and point matching.

By rearranging and then adding equation 5.14 to equation 5.16, the following representation formulas are obtained:

$$\begin{pmatrix} -\frac{\mathbf{S}^{-1}}{i\omega\mu} V^{MoM} \\ I^{int} \end{pmatrix} = \begin{pmatrix} \mathbf{Y}_{11} - \frac{\mathbf{S}^{-1}}{i\omega\mu} \left(\frac{1}{2} - \mathbf{D}\right) & \mathbf{Y}_{12} \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} \end{pmatrix} \begin{pmatrix} V^{per} \\ V^{int} \end{pmatrix} \quad (5.17)$$

where V^{MoM} is an external plane wave source (or the scattered field from another structure), and I^{int} is an internal point source.

With a known excitation, V^{int} can be obtained using the TLM algorithm, and V^{per} can be obtained at all points on the boundary using equation 5.17. I^{per} is then obtained using equation 5.16, which can then be plugged back into the TLM algorithm.

The author has presented the FD-TLM-IE to overcome the artefact resonances that are caused when exciting an object with a plane wave using purely TLM and resistive ABCs, and showed that the technique worked well to simulate accurate radiating boundary conditions. However, due to the author solving a very specific problem, one must note that equation 5.17 does not explicitly conserve the symmetry between electric and magnetic fields. The novel method described in chapter 6 includes both the EFIE and MFIE in its formulation, therefore conserving duality.

5.7 Comparisons and Conclusions

The discretised integral equations of all methods were approximated using rectangular/pulse basis functions in time and space, where the subdomains of such functions are defined by the TLM grid, and point-matching (Dirac delta testing).

All time domain hybrid methods have a coefficient in their integral equations, T , which is a consequence of following an outdated method. Newer schemes use the Gram matrix which is more advantageous to use.

The principal hybrid techniques are compared in table 5.1. Clearly, there is no particular method that is without its disadvantages, however by choosing more appropriate basis and testing functions, the schemes could have overcome many of their limitations.

A more successful method would combine the advantages of these hybrid methods, whilst avoiding their disadvantages. A superior technique should have the following characteristics:

- Solves TD fields, which allows for coupling to non-linear systems.
- Allows for easy computation of secondary technically relevant quantities such as the scattered field, radar cross sections, port impedances, and resonant frequencies.
- Provides perfectly absorbing boundary conditions that do not rely on the inaccurate TLM ABCs
- Does not need padding between objects and the TLM/BEM interface.
- Does not need to store discrete Green's functions.
- Self interactions are computed with TLM, only the coupling between objects is considered by the Green's function.
- The hybrid technique is easy to understand, and can be straightforwardly deduced from the representation theorem and continuity conditions at the inter-method boundary.
- When implementing the new technique, minimal changes should be made to the underlying methods.

The following chapter describes a method that achieves the above goals, and furthermore, also makes use of unstructured meshes to allow modelling of complex geometries using UTLM.

Table 5.1: Comparison of current hybrid TLM-BEM methods.

Method	Advantages	Disadvantages
TLM-IE	<ul style="list-style-type: none"> • More accurate open boundary condition than traditional TLM ABCs 	<ul style="list-style-type: none"> • All Green's function interactions are computed • Discretisation errors on subdomain boundaries, meaning padding between object and boundary is required
ARB	<ul style="list-style-type: none"> • TLM subdomain boundary can be as little as 1 cell from object surface • More accurate ABCs 	<ul style="list-style-type: none"> • Large number of discrete Green's functions calculated and stored • Padding of 1 cell or more is required
IRIS	<ul style="list-style-type: none"> • Simple to understand • Self interactions are computed by TLM, meaning less Green's functions are used 	<ul style="list-style-type: none"> • Necessary use of traditional TLM ABCs
TF/SF	<ul style="list-style-type: none"> • High quality PML boundaries • able to simulate externally excited fields 	<ul style="list-style-type: none"> • Complicated connection process • Scattered fields are not available in the whole domain
2D FD-TLM & EFIE	<ul style="list-style-type: none"> • More accurate ABCs 	<ul style="list-style-type: none"> • Method only proven in frequency domain with TM EFIE

References

- [5.1] L. Pierantoni, S. Lindenmeier, and P. Russer, “Efficient analysis and modelling of the radiation of microstrip lines and patch antennas by the TLM-integral equation (TLM-IE) method,” *Int. J. Numer. Model. Electron. NETWORKS, DEVICES FIELDS*, vol. 12, pp. 329–340, 1999.
- [5.2] —, “A Combination of Integral Equation Method and FD/TLM Method for Efficient Solution of EMC Problems,” *27th Eur. Microw. Conf. 1997*, vol. 2, pp. 937–942, 1997.
- [5.3] J. J. H. Wang, “Generalized Moment Methods in Electromagnetics: Formulation and Computer Solution of Integral Equations,” 1991.
- [5.4] S. Lindenmeier, L. Pierantoni, and P. Russer, “Numerical modelling of transient radiated interferences in time domain by the hybrid ARB method,” *Int. J. Numer. Model. Electron. Networks, Devices Fields*, vol. 12, no. 4, pp. 295–309, jul 1999.
- [5.5] —, “Hybrid Space Discretizing - Integral Equation Methods for Numerical Modeling of Transient Interference,” *IEEE Trans. Electromagn. Compat.*, vol. 41, no. 4, pp. 425–430, nov 1999.
- [5.6] —, “Adapted radiating boundaries (ARB) for efficient time domain simulation of electromagnetic interferences,” *1998 IEEE MTT-S Int. Microw. Symp. Dig. (Cat. No.98CH36192)*, vol. 2, pp. 465–468, 1998.
- [5.7] W. Hoefer, “The discrete time domain Green’s function or Johns matrixA new powerful concept in transmission line modelling (TLM),” *Int. J. Numer. Model. . . .*, vol. 2, no. 4, pp. 215–225, 1989.
- [5.8] S. Lindenmeier, C. Christopoulos, and P. Russer, “Methods for the modeling of thin wire structures with the TLM method,” *2000 IEEE MTT-S Int. Microw. Symp. Dig. (Cat. No.00CH37017)*, vol. 1, pp. 387–390, 2000.
- [5.9] R. Khelifi, P. Russer, and H. F. Engineering, “A hybrid method combining TLM and MoM method for efficient analysis of scattering problems,” *Microw. Symp. Dig.*, pp. 161–164, 2006.
- [5.10] M. Krumpholz and P. Russer, “A field theoretical derivation of TLM,” *IEEE Trans. Microw. Theory Tech.*, vol. 42, no. 9, pp. 1660–1668, 1994.
- [5.11] M. Naser-Moghadasi, M. Bahadorzadeh, and R. Sadeghzadeh, “Implementation of a Novel TLM-MOM Hybrid Method for the Analysis of Interference in Antennas,” *2008 3rd Int. Conf. Inf. Commun. Technol. From Theory to Appl.*, pp. 1–4, apr 2008.
- [5.12] N. Fichtner and P. Russer, “A total-field/scattered-field technique applied for the TLM-integral equation method,” *Microw. Symp. Dig. 2009. MTT’09. IEEE MTT-S Int.*, pp. 325–328, 2009.

- [5.13] A. Taflové and S. C. Hagness, *Computational Electrodynamics The Finite Difference Time Domain Method*, 3rd ed. Artech House, 2005.
- [5.14] M. Zedler and G. Eleftheriades, “Hybridisation of 2D frequency domain TLM with the MoM-discretised 2D-EFIE,” *Microw. Conf. (EuMC)*, ..., no. September, pp. 1429–1432, 2010.

The Boundary Element Unstructured Transmission-line Method

The novel hybrid method described in this thesis is called the Boundary Element Unstructured Transmission-line (BEUT) method. It is conceptually very simple and can be easily applied to existing solvers of the derived methods. In fact its derivation is directly linked to the construction of the PMCHWT integral equation for the modelling of transmission problems through piecewise homogeneous devices [6.1]. The key ingredient is the construction of a representation formula valid on the (inner) boundary of the TLM governed regions. Using the separate theories of each 2D method described previously, this chapter will detail the way in which both methods were coupled.



6.1 Theory

The hybridisation of BEM and UTLM is achieved by enforcing continuity of fields across the boundary interface.

Traditional matched boundaries for TLM consist of applying an impedance to terminate the transmission line as shown in figure 6.1a. However this does not take into account the interactions from all the other boundary edges. The BEUT method here introduces the local boundary impedance with a non-local interaction matrix resulting from the representation theorem for the external domain. This operator takes into account contributions from voltages at all boundary edges, including those from multiple surfaces, and all previous timesteps. This can be thought of as replacing the exterior region with a multi-port TLM cell connecting all boundary edge fields at the current timestep and all boundary edge fields from all previous timesteps, as illustrated in figure 6.1b.

Each link line and stub circuit model at the boundary can be reduced to a Thévenin equivalent circuit containing a voltage source and impedance, as shown in 4.12. This in turn gives rise to a single transmission line network, as shown in 6.2, where the closed circuit current and open circuit voltage can be found as

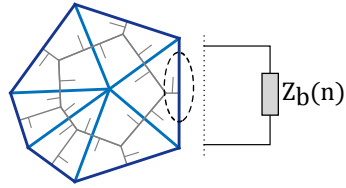
$$\begin{aligned} I_{closed} &= 2V_{link}^r Y_{link} + 2V_{stub}^r Y_{stub} \\ V_{open} &= \frac{I_{closed}}{Y_{TL}} \end{aligned} \tag{6.1}$$

where the total transmission line admittance, $Y_{TL} = Y_{link} + Y_{stub}$, and impedance $Z_{TL} = 1/Y_{TL}$.

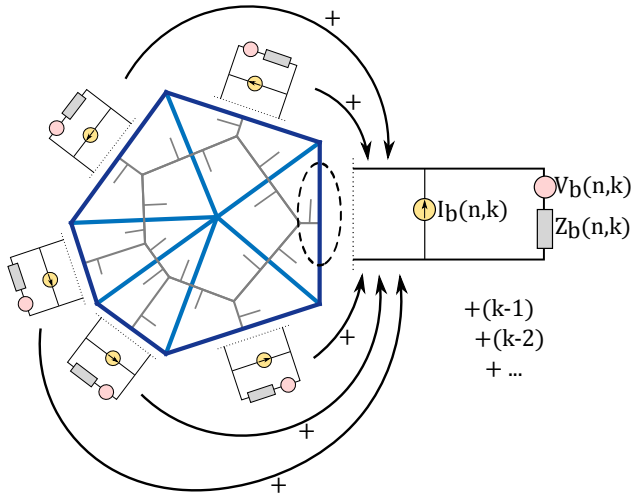
Using simple circuit analysis on the circuit shown in 6.2, the boundary voltage, V_b , and current, I_b , can be expressed as

$$\begin{aligned} V_b &= V_{open} + I_b Z_{TL} \\ I_b &= V_b Y_{TL} - I_{closed} \end{aligned} \tag{6.2}$$

With these simple equations, we can begin to couple the voltages and currents



(a) The boundary conditions used in classic TLM.



(b) The boundary conditions used in BEUT.

Figure 6.1: The circuit representations of the boundary conditions.

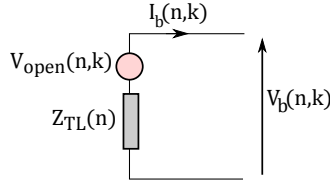


Figure 6.2: The Thévenin equivalent circuit diagram at timestep, k , for the connection at a boundary edge of triangle n .

(a.k.a. the tangential electric and magnetic fields) just inside the boundary with the tangential fields found using BEM just outside the boundary.

6.1.1 1D Comparison

Using 6.2, we can derive the one dimensional UTLM representation theorem, valid for voltages and currents on the boundaries, which reads

$$\begin{pmatrix} V_b \\ I_b \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{Z_{TL}}{2} \\ \frac{Y_{TL}}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} V_b \\ I_b \end{pmatrix} + \begin{pmatrix} \frac{V_{open}}{2} \\ -\frac{I_{closed}}{2} \end{pmatrix} \quad (6.3)$$

where subscript b denotes the boundary values, and the matrices Y_{TL} and Z_{TL} are diagonal.

The representation formulas for the equivalent boundary transmission line (eq. 6.3) is similar to the Poggio-Miller-Chan-Harrington-Wu-Tsai (PMCHWT) integral equation in one dimension. This is derived in section 2.2.2 where the final formula is shown in 2.37. In this case, the equation is evaluated only for self-patch elements, giving

$$\begin{pmatrix} e_z(\mathbf{r}, t) \\ h_t(\mathbf{r}, t) \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{\eta}{2} \\ \frac{1}{2\eta} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} e_z(\mathbf{r}, t) \\ h_t(\mathbf{r}, t) \end{pmatrix} + \begin{pmatrix} e_z^i(\mathbf{r}, t) \\ h_t^i(\mathbf{r}, t) \end{pmatrix} \quad (6.4)$$

As can be seen, the matrices in equations in 6.3 and 6.4 are identical, which underlines the link between both methods.

There is such a representation theorem for each edge on the boundary between the TLM and BEM governed domains. Arranging these in a large block diagonal system yields an interior representation theorem valid for the current time step.

6.1.2 The Boundary Element Unstructured Transmission-line Representation Formulas

The comparison of 1D BEM and UTLM means that the inner UTLM representation theorem (as shown in equation 6.3) can be equated with the exterior representation theorem (as shown in equation 2.48).

Taking into account the equivalences in 4.20 repeated in equation 6.5 for convenience), the representation formulas for the exterior domain becomes 6.6.

$$\begin{aligned} V &= \mathbf{e}_z \\ I &= \frac{\mathbf{h}_t}{l} \end{aligned} \quad (6.5)$$

$$\begin{pmatrix} V_b \\ I_b \end{pmatrix} = \begin{pmatrix} \frac{1}{2} + \mathbf{D} & -\mu \mathbf{S} l_b^{-1} \\ -\frac{\mathbf{N}}{\mu} l_b & \frac{1}{2} - \mathbf{D}' \end{pmatrix} \begin{pmatrix} V_b \\ I_b \end{pmatrix} + \begin{pmatrix} \mathbf{e}_z^i \\ l_b \mathbf{h}_t^i \end{pmatrix} \quad (6.6)$$

Taking equation 6.3 from 6.6 and rearranging slightly yields the UTLM/BEM counterpart of the PMCHWT and boundary integral equation:

$$\begin{pmatrix} \mathbf{e}_z^i \\ \mathbf{h}_t^i l_b \end{pmatrix} + \frac{1}{2} \begin{pmatrix} -V_{open} \\ I_{closed} \end{pmatrix} = \begin{pmatrix} -\mathbf{D} & \frac{Z_{TL}}{2} + \frac{\eta}{c} \mathbf{S} l_b^{-1} \\ \frac{Y_{TL}}{2} + \frac{c}{\eta} \mathbf{N} l_b & \mathbf{D}' \end{pmatrix} \begin{pmatrix} V_b \\ I_b \end{pmatrix} \quad (6.7)$$

The first term represents an exterior incident plane wave contribution and the second term represents the transmission line signals incident on the boundary.

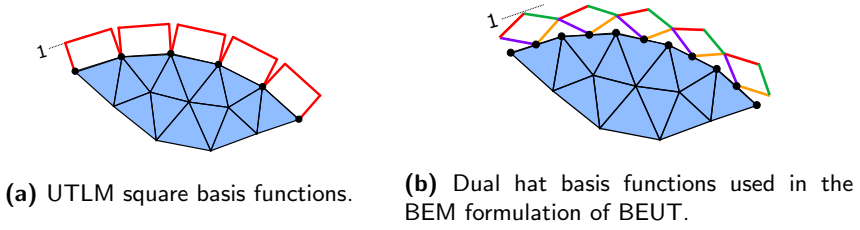


Figure 6.3: The boundary section of a scatterer showing different basis functions.

The BEUT method requires the following steps to be taken for every timestep:

- 1) Perform the UTLM scatter process, then find V_{open} and I_{closed} using equation 6.1.
- 2) Solve the coupling equation 6.7 to obtain the boundary values.
- 3) Run the UTLM connect process using the updated boundary values.

6.2 Implementation

6.2.1 Basis functions

To ensure that a stable algorithm is created and a physically reasonable timestep is chosen, Delaunay triangular meshes must be used, as explained in [6.2], where the link line lengths will never be negative. The voltages and currents that are the unknowns in the TLM description correspond to samples of the electric and magnetic field in the centers of interface edges, as shown in figure 6.3a. Therefore, to satisfy spatial continuity at the boundary edges, BEUT uses normalised dual basis functions to expand BEM field vectors which have degrees of freedoms attached to edge centers [6.3], as shown in figure 6.3b, which allows for a natural mapping between TLM and BEM degrees of freedom.

We must explain how the testing coefficient stemming from the BEM equations can be identified with the field samples (which are the degrees of freedoms)

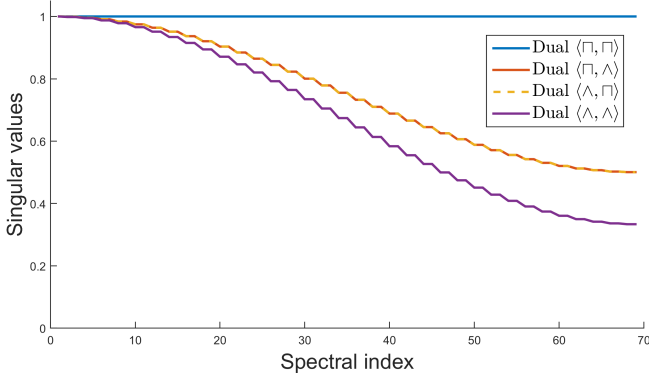


Figure 6.4: The eigen decomposition of Gram matrices for various dual basis and test functions (scaled by $1/l$) for a cylinder with 70 boundary edges.

on the TLM side. Since UTLM does not explicitly use weighting or testing functions, the BEM operators must be carefully tested to transform the normally tested unknowns into appropriate untested equivalents. This can be accomplished by making use of the inverse Gram matrix, which can be approximated to a diagonal matrix for computational purposes. To obtain a low order approximation of the inverse Gram matrix, we can scale the testing functions that make up the matrix. The scaling factor can be found by taking the eigen decomposition of the Gram matrix for different combinations of dual basis and test functions. In figure 6.4, the eigen decomposition for various dual basis and test functions of height $1/l$ is shown. We can easily see that sampling and testing with square functions in this way gives an array of singular values equal to 1. Hence, by scaling a square testing function by the inverse of the edge length that it acts upon, we can then equate the unknown with its untested counterpart.

The same reasoning can be applied to other testing functions, however now the scaling will only be valid for slowly varying fields. This constraint is already guaranteed by a rule of thumb that defines the UTLM mesh resolution, which is that each edge length should be less than a tenth of the smallest wavelength.

6.2.2 Stability

It could be argued that the unconditional stability of UTLM may be jeopardised if coupled with a conditionally stable method such as BEM. However, the optimum conditions for BEM stability can be deduced:

- 1) The stability strongly depends on the geometrical discretisation of the body, i.e. stability is more likely when the discretisation is uniform and regular, and less likely when the body has wedges or tips. [6.4]
- 2) The resolution of the integration scheme should be fine enough to resolve the discontinuities of the temporal convolutions (as depicted in figure 6.5), i.e. as $c\Delta t$ decreases, the number of Gaussian quadrature points should increase. [6.5]

For the first point, UTLM is an ideal choice for coupling with, because its meshing constraints means that the geometrical discretisation provides optimum conditions for BEM stability.

To address the second point, we must realise that BEM and UTLM share the same timestep, which is defined by UTLM and the constraint set in equation 4.31. Since this is likely to be small, the BEM temporal convolutions must be carefully integrated, especially near the convolution discontinuities and the Green's function singularities. In cases where short link lines are necessary, the timestep can be maximised by clustering ill-conditioned triangles to create higher order elements, thus increasing the link line lengths, as described in [6.2]. In our experiments, simply using a sufficient number of Gaussian quadrature points was enough to ensure stability throughout all the tests.

6.2.3 Code

An implementation using Matlab of the 2D UTLM, BEM, and BEUT methods, as described in this thesis, can be found at <https://github.com/dan-phd/BEUT>. Kernels for matrix assembly that rely upon an optimised C++/openMP

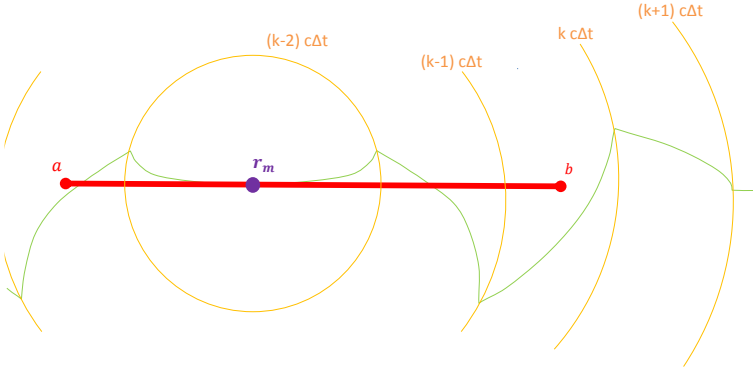


Figure 6.5: An observation point is located on an edge and has an example quadratic convolution (shown in green) at timestep k , which is to be integrated over. Discontinuities can be seen to occur at certain intervals (shown in orange).

implementation can be found at <https://github.com/dan-phd/2DTDBEM>.

Appendix A can be referred to for more information regarding the BEM, UTLM, excitation and meshing implementation in Matlab. Appendix B explains the the BEM implementation in C++, along with installation guidelines. Appendix C demonstrates how to run the compiled code step by step.

6.2.4 Parallelisation

A fast, parallelised implementation that computes the BEM operators using C++ and OpenMP has been developed to work in collaboration with the Matlab code, and can be found via the URL above.

The code was tested using a PC with 4 threads, but it soon became clear this was too slow for some of the canonical test cases, and so the code was ported and run using other computer clusters, ranging from 8 cores (within the University of Nottingham) to 36 threads (using Amazon Web Services Elastic Compute Cloud). This demonstrates the portability of the C++ code.

Unfortunately, the Matlab code running the meshing tools, UTLM algorithm and visualisation tools were not written in C++, so could not be easily optimised using parallel computing resources. The bottleneck of the project be-

came the large amount of memory needed to run the larger simulations where Matlab was needed, even with a computer with 16GB of RAM.

6.3 Alternative Methods of Hybridisation

For this project, the UTLM voltage and current values were reformulated to obtain the equivalent UTLM representation formulas that could couple easily to the BEM representation formulas.

An alternative method is to modify the BEM representation formulas to obtain values that could plug into the UTLM algorithm, which will be hypothesized here.

By rearranging the representation formula for the electric field in equation 2.48, assuming no external excitation, we can obtain

$$\mathbf{h}_t = \frac{1}{\mu} \left(D - \frac{1}{2} \right) S^{-1} \mathbf{e}_z \quad (6.8)$$

Then by plugging this into the representation formula for the magnetic field in equation 2.48, we can obtain

$$\mathbf{h}_t = \frac{1}{\mu} \underbrace{\left[\left(\frac{1}{2} - D' \right) S^{-1} \left(D - \frac{1}{2} \right) - N \right]}_{Y_{BEM}} \mathbf{e}_z \quad (6.9)$$

where Y_{BEM} can then be used as a terminating boundary admittance in the UTLM algorithm.

The issue of coupling the UTLM and BEM basis functions would be similar for this method as it is for the method described earlier. The equation in 6.9 finds the magnetic field on the boundary, directly from the electric field on the boundary. It skips finding the electric and magnetic field everywhere, which leads to an algorithm requiring a MOT inside a MOT. This could lead to slow convergence of accurate solutions, which is why the previous method was ensued.

References

- [6.1] Y. Beghein, K. Cools, F. P. Andriulli, D. De Zutter, and E. Michielssen, “A calderon multiplicative preconditioner for the PMCHWT equation for scattering by chiral objects,” *IEEE Trans. Antennas Propag.*, vol. 60, pp. 4239–4248, 2012.
- [6.2] P. Sewell, J. Wykes, T. Benson, C. Christopoulos, D. Thomas, and A. Vukovic, “Transmission-line modeling using unstructured triangular meshes,” *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 5, pp. 1490–1497, 2004.
- [6.3] K. Cools, “Mortar boundary elements for the EFIE applied to the analysis of scattering by PEC junctions,” *cccc2012 Asia-Pacific Symp. Electromagn. Compat. APEMC 2012 - Proc.*, pp. 165–168, 2012.
- [6.4] G. Manara, a. Monorchio, and R. Reggiannini, “A space-time discretization criterion for a stable time-marching\ nsolution of the electric field integral equation,” *IEEE Trans. Antennas Propag.*, vol. 45, no. 3, pp. 527–532, 1997.
- [6.5] M. Bluck and S. Walker, “Time-Domain BIE Analysis of Large Scattering Problems,” *IEEE Trans. Antennas Propag.*, vol. 45, no. 5, pp. 894–901, 1997.

Results using BEUT

In this section, the implementation of the novel BEUT method is validated and assessed in terms of accuracy and speed. Canonical test cases are demonstrated to glimpse the potential of this powerful technique.



7.1 Validation

This section will discuss the different validation techniques and results that were obtained using the BEUT method.

7.1.1 Free space cylinder excited with plane wave

To test the method initially, a 2D cylinder with a radius of 1m with the same characteristics as the background medium was meshed (as shown in figure 7.1b)

and then excited with a plane wave travelling in the positive x-direction. The expected result would be that the wave passes through without any scattering, and that the electric field seen inside the UTLM region matches the plane wave entering the region (albeit at a different time). The results are shown in figure 7.1a, and indeed match the expected results.

7.1.2 Free space cylinder excited with a point source at different locations

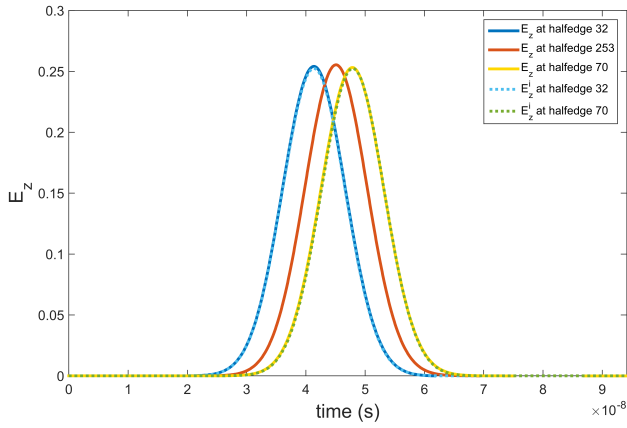
A simple test to compare the BEUT method with pure UTLM was performed using a cylinder of free space. In this test, we are using the same mesh as in section 7.1.1, but now the excitation is a point source in the interior of the cylinder, first using the traditional TLM matched impedance boundary conditions, and then repeated using the BEUT method (with the more accurate boundary conditions).

The propagation of the wave should be independent of the position of the source. Three different points in the cylinder were excited (at halfedges 58, 64 and 263), and the electric field was observed at the same relative distance away from each one (at halfedges 206, 106 and 277), as shown in figure 7.2b.

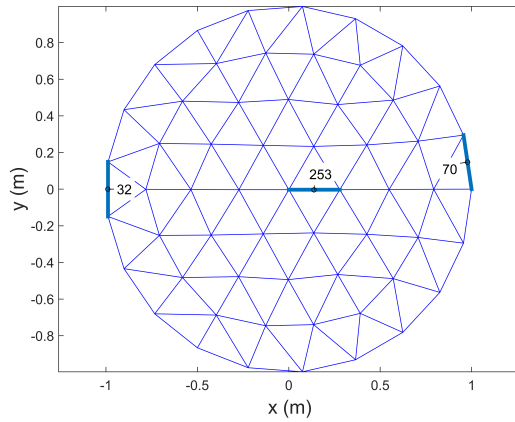
Using the TLM matched impedance boundary condition at the boundaries gives conflicting results that are polluted by spurious reflections originating from the artificial simulation domain boundary. Using the BEUT method, in contrast, gives results that are not dependent on the exact location of the source and are not affected by spurious reflection, within reasonable discretisation tolerance. The results are shown in figure 7.2a.

7.1.3 Free space cylinder excited with a point source at different frequencies

As another simple test, we excited the same meshed cylinder of free space as in section 7.1.2, but this time using a point source with different wavelengths. In this way, we can change the cutoff frequency. The higher the frequency,

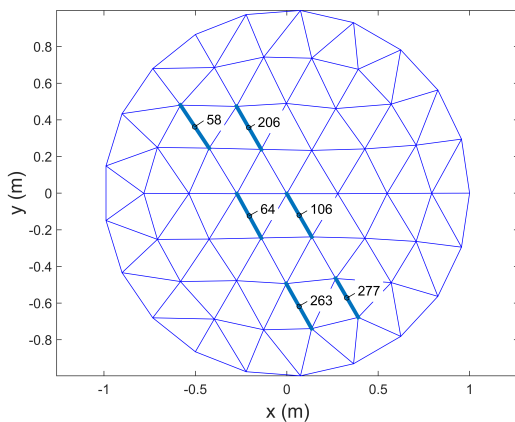


(a) Mesh showing observation points.

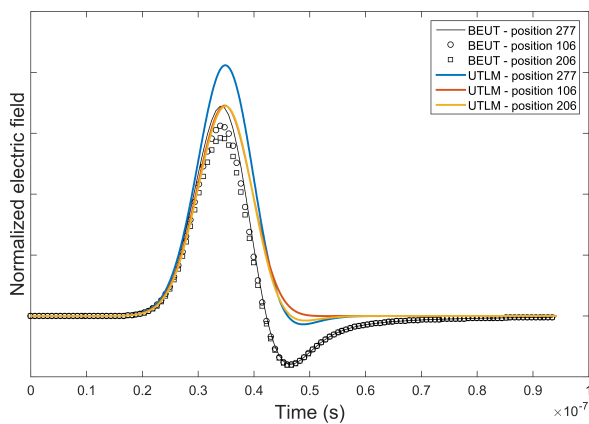


(b) Plot of time domain electric fields passing through each observation point.

Figure 7.1: Results modelling a plane wave through a cylinder of free space using the BEUT method.



(a) Mesh showing source and observation halfedges.



(b) Plot of time domain electric fields passing through each observation point.

Figure 7.2: Results comparing traditional UTLM with the BEUT method when modelling free space.

the more accurate the matched impedance boundary is and thus the results using traditional UTLM ABCs will more likely converge to the results using BEUT. This will be true up to the frequency at which dispersion error becomes significant.

The results comparing the transient response at the boundary of the cylinder for different incident waves is shown in figure 7.3a.

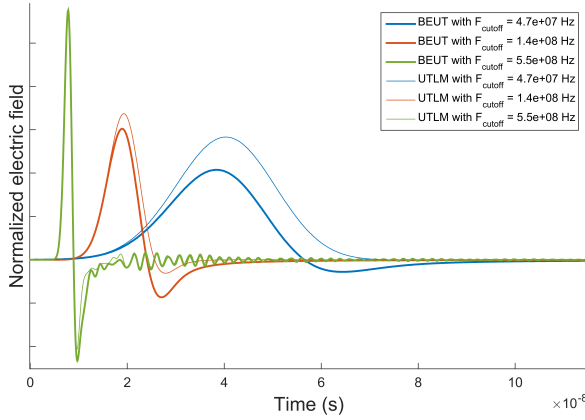
The theory that the matched impedance boundary condition is more accurate at higher frequencies is confirmed by figure 7.3b which interpolates the relative error across the frequency range that was tested.

7.2 Accuracy: Dielectric cylinder excited with a plane wave

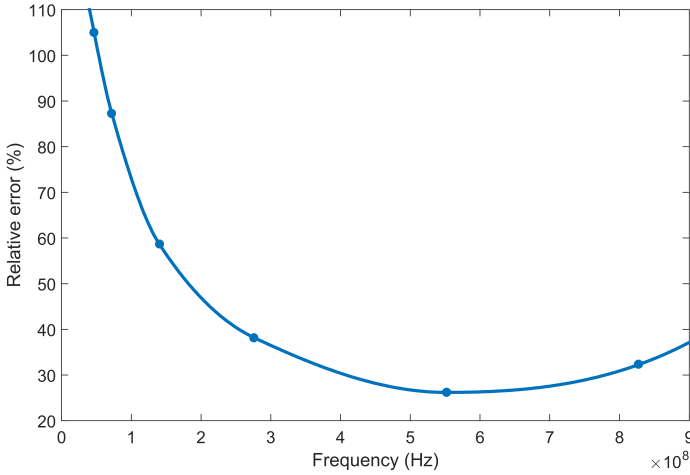
We excite a 2D dielectric cylinder meshed in a domain of free space (as shown in figure 7.4a) with a plane wave. After monitoring the electric field of the surface at the exposed side (the boundary edge nearest to the excitation source) and the shadow side (the boundary edge furthest from the excitation source) for a sufficient time, we compute the Fourier transform. There is an analytical solution to this problem in the frequency domain as can be found for example in [7.1]. This solution is compared with pure UTLM, and then with the BEUT method.

Modelling plane waves in TLM can be found for example in [7.2] and [7.3]. The boundaries normal to the angle of incidence are terminated with matched boundaries, and the boundaries tangential to the angle of incidence are terminated with open circuits. However, these boundary conditions are only accurate for the incident field, hence our UTLM model updates the boundaries once the scattered field is detected for more precise results.

The results are compared in Figure 7.4b. The frequency response using the BEUT method more closely matches the analytical results than the corresponding pure UTLM method. Of course, the areas of mesh representing free space are no longer needed if the BEUT method is used and no padding is

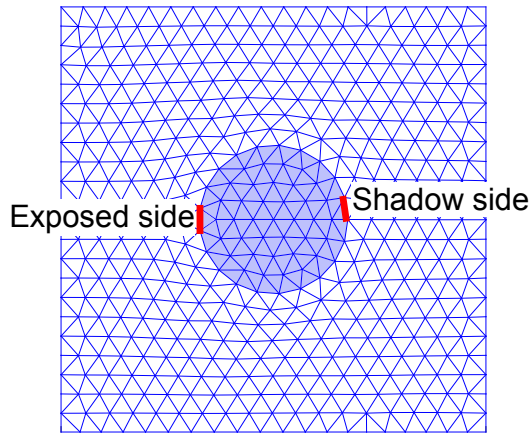


(a) Plot of time domain electric fields passing through the same point for different frequencies.

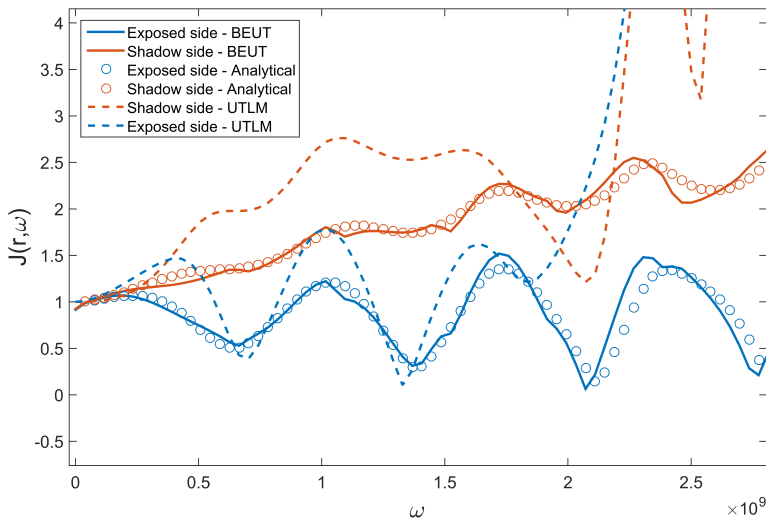


(b) Plot of the relationship between the relative error between UTLM and BEUT, and frequency.

Figure 7.3: Results comparing traditional UTLM with the BEUT method when modelling free space at different frequencies.



(a) Mesh showing observation points.



(b) Plots of frequency domain results.

Figure 7.4: Results comparing BEUT with pure UTLM and analytical solutions when modelling a dielectric cylinder.

required either, so we can simply use the inner cylindrical mesh to get even more accurate results, and with less computational resources.

7.3 Speed: Two Spatially Distinct Dielectric Cylinders

The interaction between two spatially distinct cylinders was investigated using BEUT, and the accuracy and speed was compared with a purely UTLM simulation. The UTLM mesh domain was increased (as depicted in 7.5a) to reduce the effects of the artificial absorbing boundaries, and re-compared the results using BEUT to monitor the convergence.

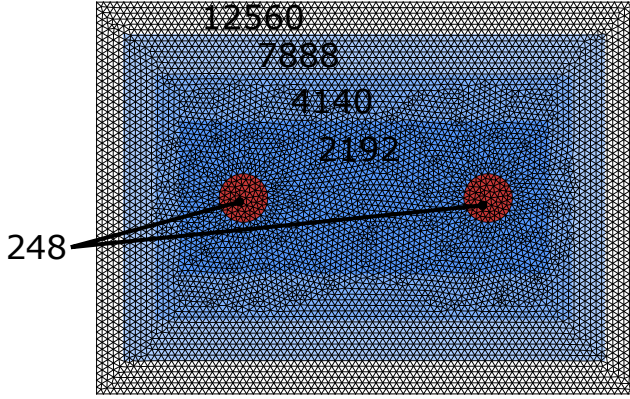
Because BEUT only requires the objects to be meshed, there were just 248 triangles to model, as opposed to pure UTLM which required the whole domain to be meshed. Consequently, the pure UTLM simulations had mesh sizes of 2192, 4140, 7888 and 12560 triangles. It is obvious to see that the more triangles that are present in the simulation, the longer the UTLM algorithm takes to compute.

As predicted, the pure UTLM results showed evidence of non-physical reflections from the matched boundaries; the beginning of which are labelled in figure 7.5b. The results from the BEUT method, however, contained no reflections and the perfectly radiating fields were observed.

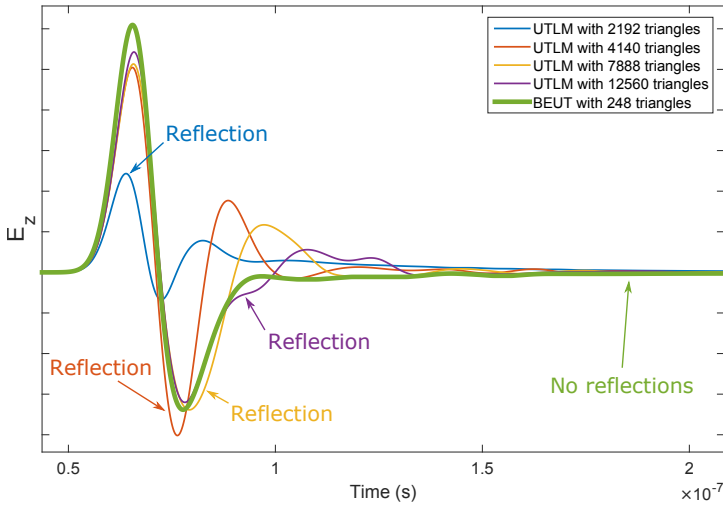
The graph in figure 7.6 shows that the UTLM results converge to the BEUT result as the mesh size increases. Figure 7.6 also reveals the times taken to run the UTLM algorithm for each test, and shows the relative speed gain when simulating less triangles.

7.4 Canonical test cases

Here we will demonstrate the suitability for problems that contain multiple smooth geometries containing inhomogeneous materials separated in free



(a) Delaunay meshes demonstrating the different sized UTLM domains surrounding the meshed cylinders, along with labels indicating the number of triangles each mesh contains.



(b) Plots of time domain results, along with labels indicating where the main reflections from imperfect boundaries begin.

Figure 7.5: Results comparing BEUT with pure UTLM when modelling two spatially distinct dielectric cylinders for different sized meshes.

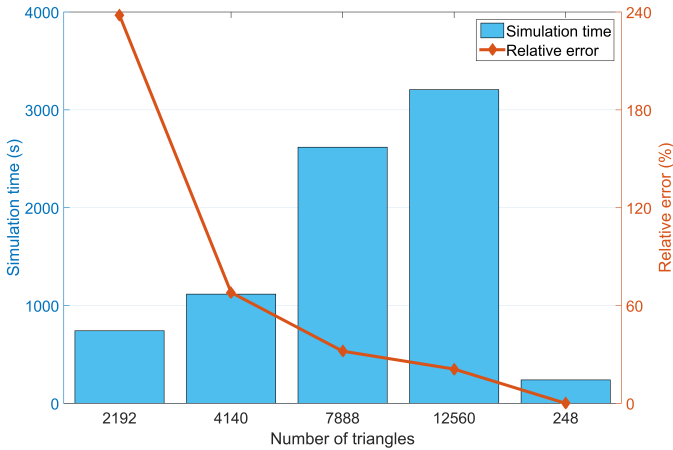


Figure 7.6: Graph showing the speed and accuracy gains of BEUT over UTLM.

space. This ability is clearly an additional advantage of the BEUT method over traditional absorbing boundary conditions.

7.4.1 Two Spatially Distinct Lüneburg Lens Antennas

To demonstrate the capability of BEUT, we modelled a point source signal transmitted over-the-air using two Lüneburg lens antennas. A Lüneburg lens antenna is a non-uniform lens that transforms a spherical wave into a plane wave (and vice versa) [7.4,7.5].

An ideal Lüneburg lens is a radially symmetric sphere with a continuously varying relative permittivity ranging from 1 at the surface to 2 at the center,

$$\epsilon_r(r) = 2 - \left(\frac{r}{a}\right)^2 \quad (7.1)$$

where a is the radius, and r is the distance from the center. Figure 7.7a shows the relative permittivity coverage and the location of the point source used in the test case.

To compute this problem using UTLM alone would be inefficient and inaccurate, as interference would occur from artificial reflections from the boundary of the simulation domain. On the other hand, BEM cannot be used to model transmission through the non-uniform Lüneburg lens. Using BEUT enables accurate modelling of the lens, and accurate modelling of the transmitted waves through free space. Furthermore, only the lenses need to be meshed, so computational resources are conserved when compared to a fully meshed solver.

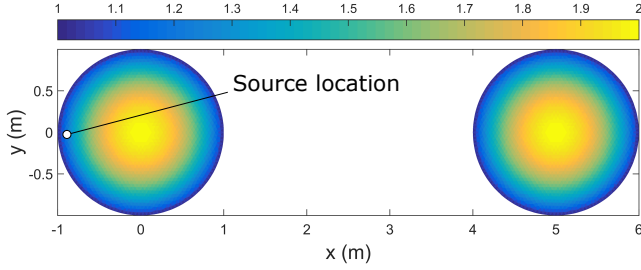
Fig. 7.7b displays the total electric field due to a point source positioned at the left lens. As can be seen, the incident wave produced by the point source is converted to a plane wave during transmission over the free space region, and then concentrated back to a point source at the right lens.

The fields inside the Lüneburg lens antennas are modelled using UTLM, and the mesh shown in fig. 7.7b is the Delaunay triangulation used in this domain. All fields outside these regions, i.e. in free space, are modelled by BEM; the mesh shown in fig. 7.7b is an auxiliary mesh used for visualisation of the fields only.

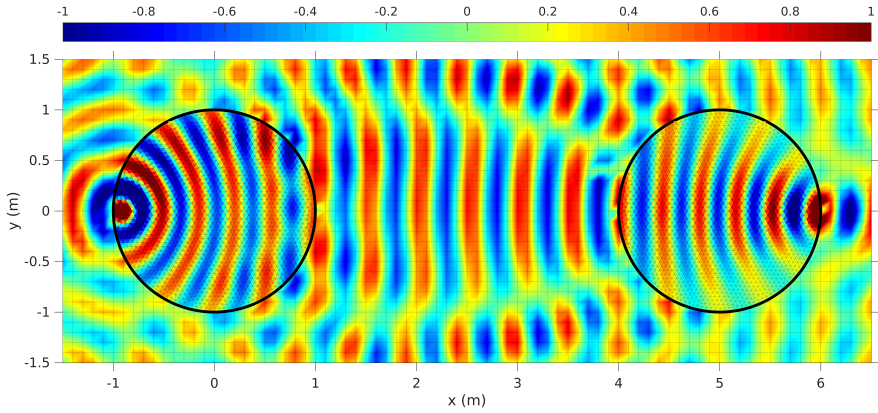
7.4.2 Dipole Antenna and Radome Interaction

As a more applied example involving an irregular geometry, the effect of a radome positioned over a radiating antenna for aerospace communication is analysed here. A radome is a structural enclosure which protects an antenna from damage by the surrounding environment without effecting performance. A typical example would be the nose of an aircraft which protects the antenna beneath from aerodynamic stresses.

The radome design is tailored to the frequency range of the protected antenna, and depends on the materials used for its construction, the number of layers, and its shape. For this analysis, we will monitor a wireless LAN dipole antenna operating at 2.5GHz, protected by a spherically blunted cone with a base inner radius of 1.2λ and an inner height of 1λ . The radome is built with a 35mm thick outer layer made of plastic polymer ($\epsilon_r = 4.8$), and a 68.7mm thick inner layer made of foam polyethylene ($\epsilon_r = 1.25$).

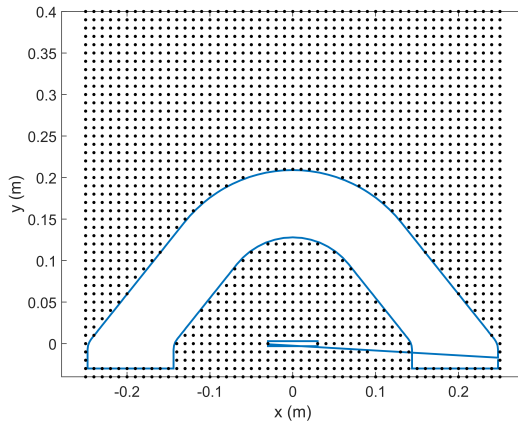


(a) Diagram showing the relative permittivity values across two Lüneburg Lenses, along with the point source location used in the test case.

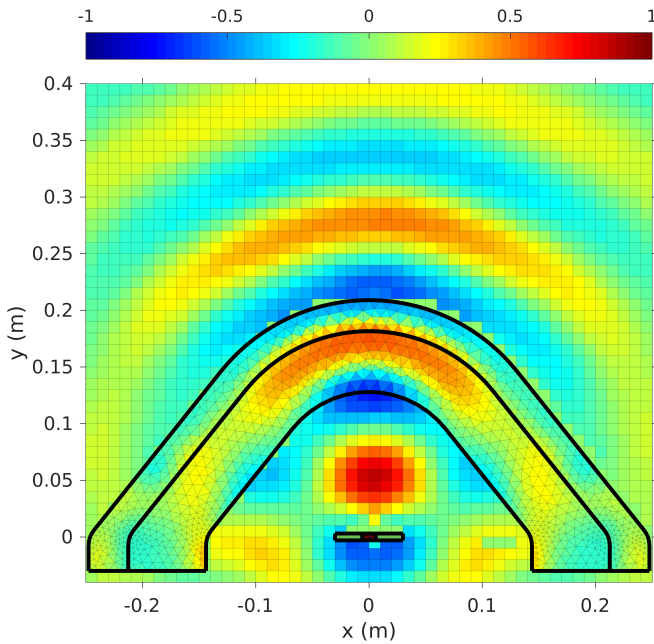


(b) 2D plot showing the total normalised electric field at a point in time.

Figure 7.7: Simulation of two spatially distinct Lüneburg lens antennas.



(a) Diagram showing the observation points used to find the scattered field in the external region.



(b) 2D plot showing the total normalised electric field at a point in time.

Figure 7.8: Simulation of the coupling between a 2.5GHz dipole antenna and a radome.

The dipole antenna consists of two identical, perfectly conducting elements either side of a free space gap where the point source is located. The antenna length is $\lambda/2$ and the structure is meshed very finely at roughly 80 edges per wavelength. The radome has a much more coarse mesh. It is common in the design of radomes to first compute the incident field radiated by the antenna, and then to model the radome using this incident field in a separate simulation, neglecting any mutual coupling. Using the BEUT method, the dome and the source geometry are modelled by a mesh with mesh size only dependent on the local geometry. This is in stark contrast to the structured TLM where the finest geometric detail determines the global mesh size. In addition, the empty space region in between dome and source is governed by the BEM, resulting in a technique that uses the bare minimum of degrees of freedom without jeopardising the solution's accuracy.

As can be seen in fig. 7.8b, the fields inside the objects are modelled using UTLM with a triangular mesh. All fields outside these regions are modelled by BEM. In this case, the scattered field is found at points defined in a structured mesh with a 10mm edge length; the auxiliary mesh is displayed in figure 7.8a. Because the vertices inside the TLM regions are discarded (for the benefit of computational resources), a function is used to determine if an observation point is inside the object. For more than one object, this function can sometimes anomalously discount a vertex (or vertices) between the objects. In this case, 3 vertices at $[0.1,0]$, $[0.11,0]$ and $[0.12,0]$ were incorrectly removed.

To confirm the effectiveness of this particular radome, we can compare the far-field of the antenna with and without the radome at the design frequency of 2.5GHz. The far-field array pattern can be obtained by measuring the electric field at a distance sufficiently far away from the source. It is directly available from the BEM boundary data and, because it is computed using the exact Green's function of the propagation environment, its accuracy is not affected by dispersion error. This comparison can be seen in fig. 7.9, where the results are normalised w.r.t. the peak antenna response.

The results show that the forward signal is amplified when the radome is used at the design frequency. This is expected because each layer of the radome has a half wavelength thickness, which introduces a 360° phase shift. Because of this phase shift, the reflections at each interface are superimposed causing

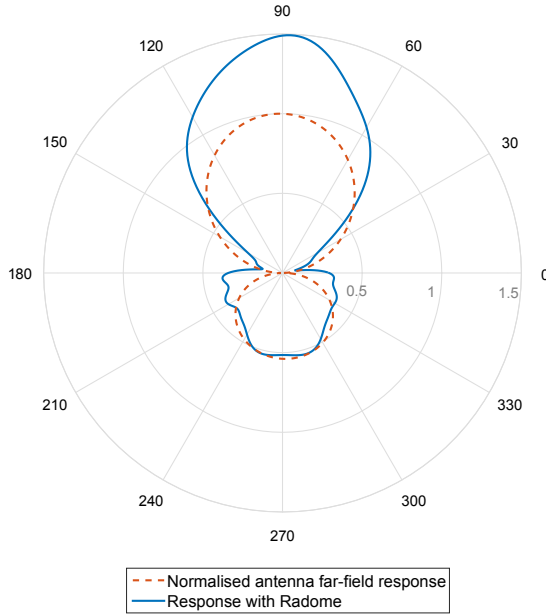


Figure 7.9: Plot showing the total normalised electric far-field response from the dipole antenna at 2.5GHz with and without the radome.

an increase in the net transmission of waves.

Computing the far field using a purely UTLM based technique would require enlarging the simulation domain and even then the far field would be compromised by spurious reflections from the simulation domain boundary and accumulated dispersion errors.

References

- [7.1] J.-M. J. Jin, *Theory and Computation of Electromagnetic Fields*, I. PRESS, Ed. John Wiley & Sons, 2011.
- [7.2] M. Zedler and G. V. Eleftheriades, “Anisotropic transmission-line metamaterials for 2-d transformation optics applications,” *Proc. IEEE*, vol. 99, no. 10, pp. 1634–1645, oct 2011.
- [7.3] L. Khashan, A. Vukovic, P. Sewell, and T. M. Benson, “Dispersion in the 2D unstructured transmission line modelling (UTLM) method,” in *2015 IEEE Int. Conf. Comput. Electromagn.* IEEE, feb 2015, pp. 347–349.
- [7.4] H. Mosallaei and Y. Rahmat-Samii, “Nonuniform Luneburg and two-shell lens antennas: radiation characteristics and design optimization,” *IEEE Trans. Antennas Propag.*, vol. 49, no. 1, pp. 60–69, 2001.
- [7.5] M. Bosiljevac, M. Casaletti, F. Caminita, Z. Sipus, and S. MacI, “Non-uniform metasurface luneburg lens antenna design,” *IEEE Trans. Antennas Propag.*, vol. 60, no. 9, pp. 4065–4073, 2012.

Conclusions

This chapter of the thesis will revisit the significant segments of the project along with some concluding statements. Some proposed directions for future work in this area will also be suggested, including the ideal steps needed for the inevitable expansion to 3D.



8.1 Overview of the work presented

The aim of this work was to provide and demonstrate a new technique that could model the electromagnetic scattering between non-homogeneous, possibly non-linear objects with geometrically complex features.

This thesis has presented the 2D BEUT method, a novel EM simulation technique which hybridises the BEM and UTLM methods. The individual techniques were derived, and implementation guidelines were described in de-

tail.

The novel technique combines the power of UTLM (unconditional stability, ability to model inhomogeneous materials and smooth geometries) with the accuracy of TD BEM (perfectly radiating boundaries, resolvable fields anywhere in free space). The technique can find the scattered field anywhere in the domain, provides perfectly absorbing boundary conditions that do not rely on the inaccurate TLM ABCs, does not need padding between objects and the TLM/BEM interface, and does not need to store discrete Green's functions. Moreover, the technique is easy to understand, and minimal changes are required to the underlying methods for implementation. These advantages differentiate the method from previous hybridisation attempts, and make it more favourable to use when modelling transient scattering problems involving large free space regions where accuracy is vital.

The technique was successfully demonstrated by modelling non-trivial structures within free space domains. The new technique outperformed UTLM in comparative simulations in terms of accuracy, and it showed how perfectly radiating boundaries can be used in UTLM. Furthermore, BEUT reduced the number of meshed elements when compared with pure UTLM, thus reducing the computational resources required when performing the scatter and connect processes.

Simple test cases showed significant accuracy and speed gains compared to using pure UTLM. Scattering between two spatially distinct, non-uniform Lüneburg lenses, and also the scattering between an antenna and a radome were shown using BEUT. Results obtained from the demonstrations matched that of expected and previously published results.

There is enormous scope for the BEUT method. In space applications, the accurate response due to radiation from multiple satellites at far distances away can now be modelled without needing to mesh the free space in between. Recent advances in TLM also allow it to model materials with frequency dependent properties [8.1], which would be useful for measuring the effects of solar wind, and spacecraft charging. The method would be equally valuable to perhaps the electronics, communications, automotive, military, and scientific engineering industries, and consequently, the overall benefit is very signifi-

cant.

To summarize, this thesis has described the development of a new simulation technique which can provide more accurate and efficient solutions to transient scattering problems involving complex materials and geometries which are immersed in free space.

8.2 Future work

In this contribution, we use a straightforward implementation of the 2D BEM which could be further accelerated using techniques such as the Time Domain Adaptive Integral Method (TD-AIM) [8.2], and the Plane Wave Time-Domain (PWTD) algorithm [8.3].

For BEUT to fully realise its potential, the extension to 3D needs to be achieved. As well as the advantage of modelling more realistic scenarios, the 3D BEM uses a Green's function which does not have a tail (unlike the 2D Green's function), thus the MOT algorithm will perform faster as it does not depend on *all* previous timesteps.

However, the basis functions commonly associated with 3D BEM are the Rao-Wilton-Glisson (RWG) functions [8.4], which have degrees of freedoms attached across the edges, whereas 3D UTLM has degrees of freedoms at the center of faces; this may introduce difficulties in coupling the 3D techniques. Initially, it may be worth attempting to interpolate the BEM surface currents to a piecewise constant value for each face which could then assist in coupling to the UTLM unknowns. Failing this, there may be an equivalent method for the 3D case which is similar to the dual basis functions that were used in the 2D case to match the degrees of freedoms. Another option would be to refer to [8.5–8.8] and implement a penalty (Nitsche-type) method so that a discontinuous Galerkin scheme can be used.

References

- [8.1] J. Paul, C. Christopoulos, and D. Thomas, “Generalized material models in TLM .I. Materials with frequency-dependent properties,” *IEEE Trans. Antennas Propag.*, vol. 47, no. 10, pp. 1528–1534, 1999.
- [8.2] W. C. Chew, J.-M. Jin, E. Michielssen, and J. Song, *Fast and Efficient Algorithms in Computational Electromagnetics*. Artech House, Inc., 2001.
- [8.3] B. Shanker, A. Arif Ergin, K. Aygün, and E. Michielssen, “Analysis of transient electromagnetic scattering phenomena using a two-level plane wave time-domain algorithm,” *IEEE Trans. Antennas Propag.*, vol. 48, no. 4, pp. 510–523, 2000.
- [8.4] S. Rao, D. Wilton, and A. Glisson, “Electromagnetic scattering by surfaces of arbitrary shape,” *IEEE Trans. Antennas Propag.*, vol. 30, no. 3, pp. 409–418, may 1982.
- [8.5] P. Hansbo and M. G. Larson, “Discontinuous Galerkin methods for incompressible and nearly incompressible elasticity by Nitsche’s method,” *Comput. Methods Appl. Mech. Eng.*, vol. 191, no. 17-18, pp. 1895–1908, 2002.
- [8.6] G. N. Gatica, N. Heuer, and F.-J. Sayas, “A direct coupling of local discontinuous Galerkin and boundary element methods,” *Math. Comput.*, vol. 79, no. 271, pp. 1369–1394, jan 2010.
- [8.7] Z. Peng, K.-H. Lim, and J.-F. Lee, “A Discontinuous Galerkin Surface Integral Equation Method for Electromagnetic Wave Scattering From Nonpenetrable Targets,” *IEEE Trans. Antennas Propag.*, vol. 61, no. 7, pp. 3617–3628, jul 2013.
- [8.8] D. Dault and B. Shanker, “An Interior Penalty Method for the Generalized Method of Moments,” *IEEE Trans. Antennas Propag.*, vol. 63, no. 8, pp. 3561–3568, aug 2015.



BEUT Matlab Implementation Manual

The implementation for BEUT can be found at <https://github.com/dan-phd/BEUT>. The Matlab code has the ability to create meshes, compute 2D BEM, UTLM and BEUT, and output graphs, figures and animations. However, computing the BEM kernels is very slow and takes a large amount of memory. For scenarios that are more complex than a cylinder *, you will need to use an external mesher and the C++ implementation to compute the BEM operators.

*A cylinder with edges of equal length has the special advantage of having identical diagonal elements in the BEM operator matrices, meaning that a "cheat" can be performed whereby only 1 row and column needs to be computed for each timestep - significantly improving the run-time speed.

The BEUT project repository has the following directory structure:

```

BEUT .....license and readme files
├── +BEUT .....project folder
│   ├── +BEM .....container for the BEM solver
│   │   ├── +Analytical .....analytical solutions for comparison
│   │   ├── +Demo .....examples demonstrating BEM components
│   │   └── +Main .....main code for BEM test cases
│   ├── +Excitation .....container for the excitation classes
│   │   ├── +Demo ....examples demonstrating the excitation classes
│   │   └── +Main .....main code for the BEUT test cases
│   ├── +Meshing .....container for the meshing classes
│   │   ├── +distmesh .....a third party mesher
│   │   ├── +Main .....main code for creating or loading meshes
│   │   ├── meshes .....contains .mat meshes ready to be used
│   │   │   └── unconverted .....contins meshes to be converted
│   └── +UTLM .....container for the UTLM solver
│       ├── @UTLMClass .the UTLM class definition and its methods
│       ├── +Analytical .....analytical solutions for comparison
│       └── +Main .....main code for UTLM test cases

```

To install the project in Matlab, place the entire contents of **+BEUT** inside a Matlab search path.

The **+** character at the beginning of a folder defines a package folder in Matlab, which allows better organisation of classes and functions. The **@** character at the beginning of a folder defines a class folder in Matlab, which allows the use of multiple files for one class definition. These characters are not used when calling package members; for instance, creating an instance of **UTLMClass** is performed using the following syntax:

```
BEUT.UTLM.UTLMClass()
```

The **+BEUT/+Main** folder contains examples on how to use BEUT. There are **+Main** folders elsewhere in the project to demonstrate the use of UTLM and BEM as individual solvers. The **+BEUT/+Meshing/+Main** folder is the first point of call for creating a mesh or loading a custom 2D mesh from an external file. For demonstration and testing of individual classes and functions, refer

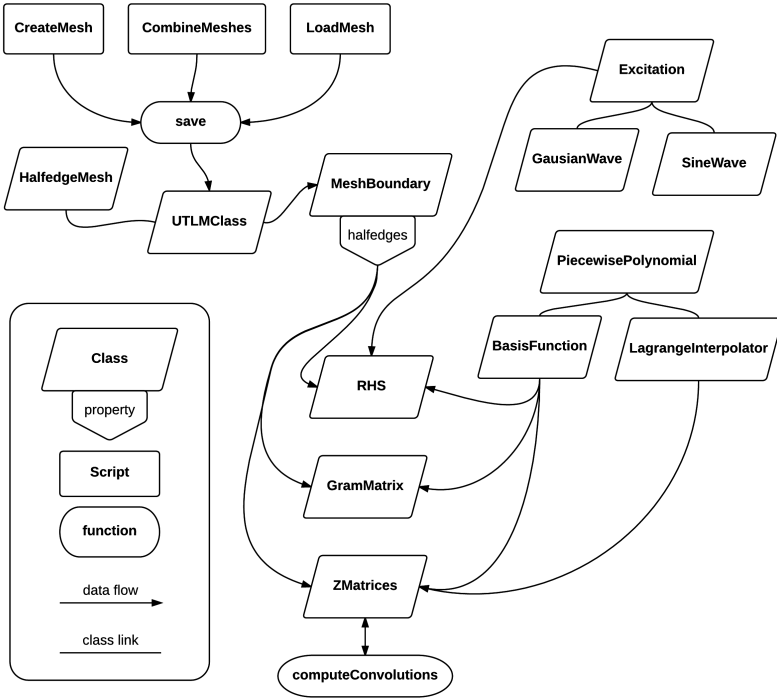


Figure A.1: The flow diagram showing the link between the different components of the Matlab BEUT implementation.

to the `+Demo` folders.

For convention, script and class names begin with a capital letter, functions begin with a lower case letter (unless an acronym is used). Generally, you will only want to open and run scripts in the `+Main` and `+Demo` folders.

If using the C++ program to compute BEM operators, you will need to change the path which stores the resulting `.mat` files; This can done by modifying the path string in `+BEUT/CFolder.m`.

The following sections will describe the major functions, classes and scripts that can be found in the BEUT project, which are linked as shown in the diagram of figure A.1.

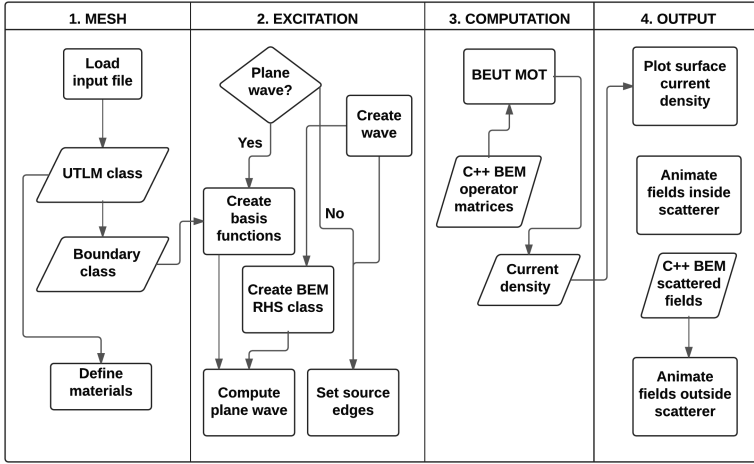


Figure A.2: The process diagram of the meshing implementation.

A.1 BEUT

The general flow of a typical BEUT simulation is shown in figure A.2, which can be summarised as follows:

- 1) Create/load mesh and save as a `UTLMClass` object (for UTLM) and a `MeshBoundary` object (for BEM), and then set material parameters.
- 2) Create excitation (as either a point source or plane wave).
- 3) Calculate BEM operators (using either the Matlab or C++ solver), and then roll through the timestepping loop (Marching-on-in-Time).
- 4) View results by plotting the surface currents, or animating the fields inside the scatterer. Further computation can be done to find the fields anywhere outside the scatterer, or even animate an entire region of space.

Appendix C demonstrates these steps for a typical simulation example.

The main scripts inside the `+BEUT/+Main` folder are as follows:

ModelFreeSpace1 (script)

In this script, the results demonstrated in section 7.1.1 are demonstrated. Region 1 is a UTLM cylinder of free space, region 2 is the external region of free space. The cylinder is excited with a plane wave. Note that this script is the only one in **+BEUT/+Main** which allows you to choose whether to use Matlab to compute the BEM operators, the others will require using the C++ program.

ModelFreeSpace2 (script)

In this script, the results demonstrated in section 7.1.2 are demonstrated. Region 1 is a UTLM cylinder of free space, region 2 is the external region of free space. The cylinder is excited with a point source at different locations, and then the results are compared using pure UTLM.

ModelDielectric (script)

In this script, the results demonstrated in section 7.2 are demonstrated. Region 1 is the meshed background of free space, region 2 is the inner dielectric cylinder. The domain is excited with a plane wave, and then the results using BEUT and pure UTLM are compared against the analytical solutions in the frequency domain.

ModelLuneburgLens (script)

In this script, a Lüneburg lens (as described in section 7.4.1) is modelled. Region 1 is the densely meshed Lüneburg lens which is excited with a point source. An animation is shown for the fields inside the lens, and optionally, outside the lens once complete.

A.2 Meshing

The top of the diagram in figure A.1 demonstrates the link between the different components of the meshing code in Matlab.

The major elements related to the meshing part of the project are as follows.

CreateMesh (script)

In this script, the library *distmesh* (found at <http://persson.berkeley.edu/distmesh>) is used inside Matlab to create 2D meshes ready for simulation purposes. However, the tool is only useful for meshing simple shapes, for example a circle or square. For more complex structures, a dedicated mesher should be used, where the resulting file can be imported into Matlab. A good 2D mesher is featured in the Comsol Multiphysics® modeling software (uk.comsol.com) where meshes can be saved as a *.mphtxt* file and imported to the Matlab project. Alternatively, there are free 2D meshers in the form of Triangle (www.cs.cmu.edu/~quake/triangle.html) and Gmsh (gmsh.info) which also have the ability to export files suitable for use in the Matlab project.

CombineMeshes (script)

This script allows the user to load two meshes and combine them so one is spatially distinct from the other.

LoadMesh (script)

This script loads a mesh from an external file, converts it to a halfedge mesh format (an instance of *UTLMClass*) that the project can use, sets the material ID numbers for each triangle (if necessary), and then checks the mesh to make sure there are no excessively small link lines.

The supported input file types in terms of their extensions are:

- *.in* (custom)
- *.gmsh/.msh* (GMSH)
- *.mphtxt* (Comsol)
- *.obj* (Wavefront)
- *.node* along with the corresponding *.ele* (Triangle)
- *.mat* (BEUT)
- *.poly* (Triangle)
- *.gid* (GiD)

save (function)

This function takes the halfedge mesh as an input and saves it to a file ready for Matlab simulation. It also extracts the mesh boundary and

saves this in a file ready for the C++ BEM operator computation.

Arguments

- **mesh** - UTLMClass object.
- **dual** - boolean to decide whether to use dual basis functions.
- **mat_file** - the path and filename to output the file that Matlab will use. This can be an empty string if not required.
- **c_file** [optional] - the path and filename to output the file that C++ will use. This can be an empty string if not required.
- **dt** [optional] - the chosen timestep. This will be calculated automatically if not given.
- **mu_0** [optional] - the permeability of free space. This will be calculated automatically if not given.
- **eps_0** [optional] - the permittivity of free space. This will be calculated automatically if not given.

HalfedgeMesh (class)

This class acts as a database for the halfedge mesh. A halfedge is simply the side of the edge belonging to a particular face.

Properties

- **vertices** - a [*number of vertices* \times 2] matrix with the first column indicating the x-coordinate and the second column indicating the corresponding y-coordinate of each vertex.
- **material_boundaries** - an array of cells, one for each material. Inside each cell contains an array of halfedge indices that act as the boundary for that material.
- **mesh_boundary** - an array of halfedge indices that act as the boundary of the whole mesh. The halfedges are connected when the array is read in order.
- **mesh_body** - an array of halfedge indices that are inside the mesh and not on the boundary.
- **faces** - an array of structs. Each struct contains the **vertices**, **fnum**, and **area** for the corresponding face index.
- **halfedges** - array of structs. Each struct contains information on

the associated halfedge, including:

- **face**
- **vertices**
- **flip** (the flip halfedge index)
- **circumcenter**
- **midpoint**
- **edgeLength**
- **linkLength**
- **TR** - Matlab triangulation object - stores **Points** and **ConnectivityList**.
- **nF** - number of faces.
- **nV_face** - number of vertices per face. For 2D triangles, it will always be 3.
- **nH** - number of halfedges.
- **num_materials** - number of different materials used in the mesh.
- **shortestLinkLength** - the shortest link length.
- **color** - an array for cycling through material colors (when plotting).

Constructor arguments

- **vertices** - as above.
- **faces** - a $[number\ of\ faces \times 3]$ matrix with each row containing 3 vertex indices that are connected to form a triangle.
- **fnum** [optional] - an array of size $[number\ of\ faces]$ with each element defining the face number of the triangle (in an iterative manner), which will be later used to decide the material placement. If the mesh contains just 1 material, the array would be full of 1s.

Methods

- **plot_mesh** - plots the mesh and renders each triangle depending on the material number it represents.
- **plot_face(face)** - plots the mesh and labels the faces given in the array **face**. If no argument is given, all the faces are labelled.
- **plot_vertex(vertex)** - plots the mesh and labels the vertices given in the array **vertex**. If no argument is given, all the vertices are labelled.
- **plot_halfedge(he,col)** - plots the mesh and labels the halfedges

given in the array **he**. Each labelled halfedge is highlighted with a color indicated by **col**. If **col** is not given, all halfedges are colored the same. If **he** is not given, all the halfedges are labelled.

- **plot_boundary** - plots and labels the halfedges that act as boundaries to different materials.

MeshBoundary (class)

This class acts as a database for the boundary halfedge mesh used by BEM.

Properties

- **halfedges** - array of structs. Each struct contains information on the associated boundary halfedge, including:
 - **he_idx** (halfedge index that this boundary edge corresponds to in the original halfedge mesh from which this class is derived)
 - **a** (vertex at the head of the edge)
 - **b** (vertex at the tail of the edge)
 - **l** (length of the edge)
 - **t** (vector tangent to the edge)
 - **n** (vector normal to the edge)
 - **shape** (shape number that this edge belongs to)
- **dual** - array of structs which has the same fields as **halfedges** but is used as an alternative to **halfedges** when using dual basis functions.
- **n_V** - number of vertices (equal to the number of halfedges for a closed surface).
- **num_shapes** - number of distinct shapes in the mesh.
- **color** - an array for cycling through material colors (when plotting).

Constructor arguments

- **HalfedgeMesh** - as above.

Methods

- **plot(he)** - plots the mesh and labels each halfedge given in the

array **he** If no argument is given, all halfedges are labelled.

A.3 Excitation

An excitation is generally a time dependant function which acts as a wave that can then be applied to the algorithm of choice, either as a point source (injected directly to the source) or as a plane wave (which can be used after appropriate testing).

We have implemented two types of function, the Gaussian pulse as shown in figure A.3a, and the sinusoidal signal. The sinusoidal wave can be modulated by a Gaussian pulse, as shown in figure A.3b, or a cosine wave can be used to envelope just the beginning and end of the signal, as shown in figure A.3c.

The major elements related to the excitation section of the project are as follows.

Excitation (abstract class)

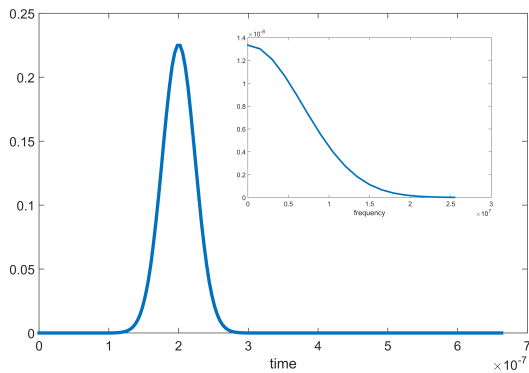
This class acts as a container for any class that computes a function to be used as an excitation.

Properties

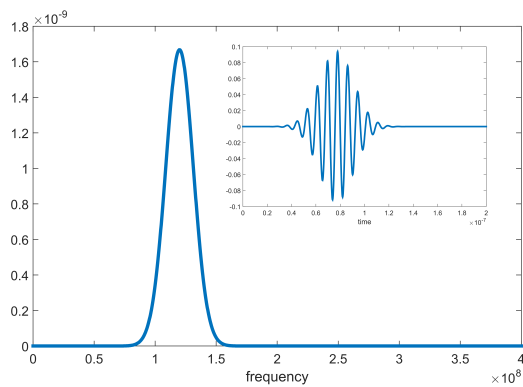
- **c** - speed of wave propagation.
- **direction** - row vector specifying direction as a 2D unit vector; e.g. $[1 \ 0]$ = x-direction, $[0 \ -1]$ = negative y-direction, 1 = normal to plane (z-direction).
- **T** - width of pulse.
- **t0** - time of arrival (as a ratio of T).
- **A** - a factor by which to scale the wave amplitude.

Methods

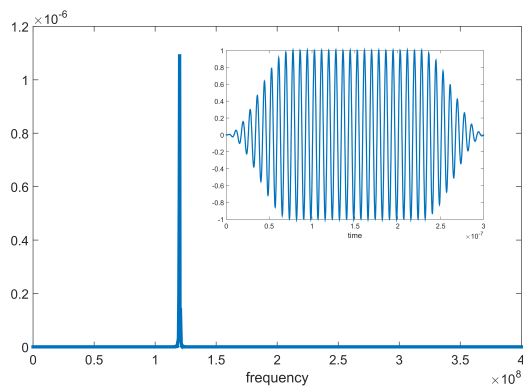
- **Fc = freq_response(time_array,plot_fig)** - determine the fre-



(a) Time domain Gaussian pulse with corresponding frequency domain inset.



(b) Frequency domain of a sinusoidal wave modulated by a Gaussian pulse, with corresponding time domain inset.



(c) Frequency domain of a sinusoidal wave modulated by a cosine wave, with corresponding time domain inset.

Figure A.3: Plots of different excitation functions.

quency response of the wave where the `time_array` as an array of time values at which to evaluate the function at. The optional boolean `plot_fig` specifies whether or not to plot the frequency domain (default is `false`). The output `Fc` is the cutoff frequency (the frequency at which the spectrum reaches below 1% of its peak).

GaussianWave (subclass of **Excitation**)

Properties

All properties are inherited from **Excitation**.

Constructor arguments

- `width` - width of pulse.
- `timeOfArrivalRatio` (optional) - time of arrival (as a ratio of `width`). Default is 1.5.
- `c` (optional) - speed of wave propagation. Default is 1.
- `direction` (optional) - row vector specifying direction as a 2D unit vector. Default is `[1 0]` (which specifies the positive x-direction).

Methods

- `E = eval(t, rho)` - evaluates the function at time `t`, at location `rho` (distance from source). If `rho` is not given, `E` is evaluated at `rho=0`.
- `E = evalDifferential(t, rho)` - evaluates the differential of the function at time `t`, at location `rho` (distance from source). If `rho` is not given, `E` is evaluated at `rho=0`.
- `E = evalIntegral(t, dt, rho)` - evaluates the integral of the function at time `t` with timestep `dt`, at location `rho` (distance from source). If `rho` is not given, `E` is evaluated at `rho=0`.
- `E = evalAmplitudeResponse(omega)` - evaluates the amplitude response due to the time domain signal at the frequency given by `omega`.

SineWave (subclass of **Excitation**)

Properties

As well as the properties inherited from `Excitation`;

- `f` - value of the modulated frequency.
- `envelope` - string indicating the envelope to modulate the wave. This can be “Gaussian” or “cosine”.
- `G` - Gaussian pulse object (to be used for the envelope).

Constructor arguments

- `freq_width` - width of pulse (in the frequency domain).
- `modulated_frequency` - frequency of arrival (in the frequency domain).
- `c` (optional) - as above.
- `direction` (optional) - as above.
- `timeOfArrivalRatio` (optional) - time of arrival (as a ratio of whatever the time of arrival is calculated at inside the constructor). Default is 1.5.

Methods

- `E = eval(t, rho)` - evaluates the function at time `t`, at location `rho` (distance from source). If `rho` is not given, `E` is evaluated at `rho=0`.

A.4 UTLM

The diagram in figure A.4 demonstrates the link between the different components of the code in a typical UTLM simulation.

The major elements related to the UTLM section of the project are as follows.

`UTLMClass` (subclass of `HalfedgeMesh`)

This class acts as the storage and computation for all UTLM variables

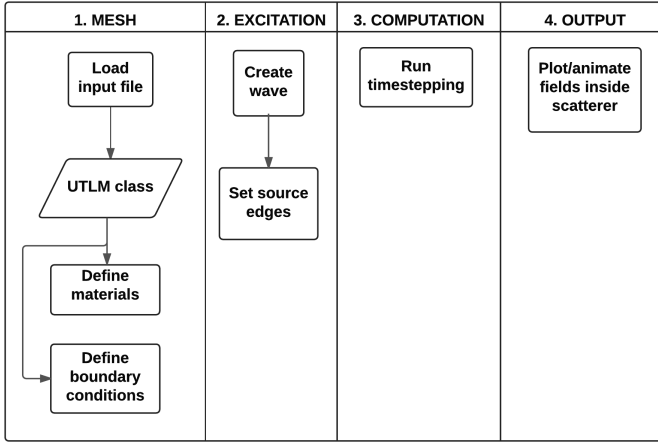


Figure A.4: The process diagram of the UTLM implementation.

and functions.

Properties

As well as the properties inherited from `HalfedgeMesh`;

- **fields** - a struct which contains the following matrices, each matrix is of size $[number\ of\ halfedges \times N_T]$ and represent the fields at all halfedges for all timesteps:
 - **E_z** (z-directed electric field)
 - **H_xy** (magnetic field tangential to the plane)
 - **H_x** (x-directed magnetic field)
 - **H_y** (y-directed magnetic field)
- **V0** - a matrix of size $[number\ of\ faces \times N_T]$ which represents the voltage at all triangle circumcenters for all timesteps.
- **I0** - a matrix of size $[number\ of\ faces \times N_T]$ which represents the current at all triangle circumcenters for all timesteps.
- **dt** - timestep value.
- **reflection_coeff** - an array of size $[number\ of\ halfedges]$ which indicates the reflection coefficient for each halfedge. This also includes halfedges inside the object because PEC boundaries may need to be considered.
- **Y_boundary** - an array of size $[number\ of\ halfedges]$ which indicates

the boundary admittance for each halfedge.

- **PEC_boundary** - an array of variable size which indicates the halfedges that are to reflect signals because of a PEC boundary.
- **V_open** - an array of size [*number of boundary halfedges*] which is used to store the open circuit voltage at the boundary halfedge for the current timestep.
- **I_closed** - an array of size [*number of boundary halfedges*] which is used to store the closed circuit current at the boundary halfedge for the current timestep.
- **halfedges** - array of structs. Each struct contains information on the associated halfedge, including the fields defined in **HalfedgeMesh**, plus:
 - **V_linki** (incident link voltage for the current timestep)
 - **V_linkr** (reflected link voltage for the current timestep)
 - **V_stub** (stub voltage for the current timestep)
 - **doConnect** (boolean to determine if the halfedge is to be required in the connect process)
 - **Y_link** (link line admittance associated with the halfedge)
 - **Y_stub** (stub line admittance associated with the halfedge)

Constructor arguments

The constructor is inherited from **HalfedgeMesh**.

Methods

- **excite_E(sourceEdges, V_source)** - excite all source halfedges specified by **sourceEdges** with the value given by **V_source**.
- **animate(field, scale)** - animate either the electric field (using **field** = 'E') or magnetic field (using **field** = 'H'), where the field is normalised and **scale** allows the user to empirically scale the fields so that the colorbar gives a clearer result. This is sometimes required because UTLM usually has a much larger field at the excitation than everywhere else.
- **calcAdmittance** - calculate admittances and set connect flags.
- **checkMesh** - plot Voronoi diagram and highlight the location of the shortest link length. The text output to screen tells the user the shortest link length, the average link length, and the ratio (which

from experience is good below 10, and best below 5).

- **connect(k, E)** - run the connect process for timestep **k**. If the electric field **E** is given, then we use this as the incident voltage at the boundary halfedges, as is stipulated by the BEUT method. If **E** is not given, compute the connect process as normal.
- **plot_interfaces** - plot the mesh interfaces between different materials.
- **plot_materials(material_parameter)** - plot the mesh with the color of each triangle equal to the values of relative permittivity (using **material_parameter = 'eps_r'**) or relative permeability (using **material_parameter = 'mu_r'**).
- **reset** - re-initialise all voltages.
- **scatter(k)** - run the scatter process for timestep **k**.
- **setBoundary(condition)** - set the boundary condition for the whole domain, or for each boundary halfedge individually. Each **condition** can be 1(=open), -1(=short), 0(=absorbing).
- **setMaterial(relative_eps, relative_mu)** - set the relative permittivity (**relative_eps**) and relative permeability (**relative_mu**) of the mesh. Each argument can either be a value that should be applied across the entire mesh, an array of size [*number of faces*], or an array of size [*number of materials*].

A.5 BEM

The diagram in figure A.5 demonstrates the link between the different components of the code in a typical BEM simulation.

The major elements related to the BEM part of the project are as follows.

PiecewisePolynomial (class)

This class acts as a superclass for any piecewise polynomial function.

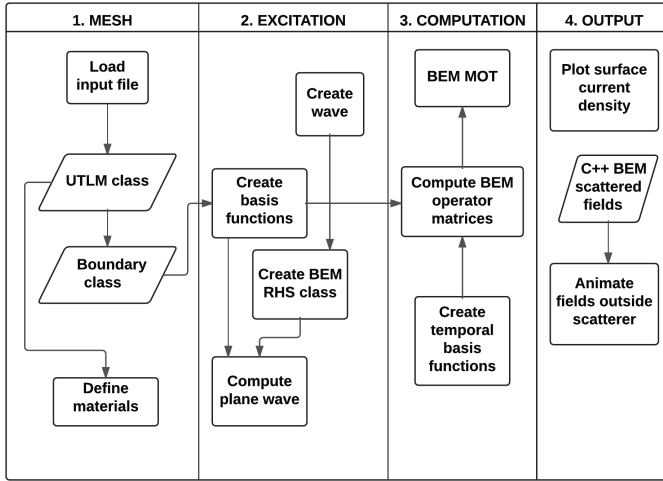


Figure A.5: The process diagram of the BEM implementation.

Properties

- **degree** - the maximum degree of all the polynomials.
- **coeffs** - a matrix of $[size\ number\ of\ partitions \times degree+1]$ where each row defines the coefficients of the polynomial for that partition.
- **partition** - an array of $[size\ number\ of\ partitions+1]$ which determines the limits of each polynomial.

Constructor arguments

- **partition** - as above.
- **coeffs** - as above.
- **degree** - as above.

Methods

- **y = eval(t)** - evaluate the polynomial at a point or points in time **t**, to output **y** of the same size as **t**.
- **obj = translate(k, p)** - copy the *PiecewisePolynomial*, shift the copied function across the x-axis by the value of **k**, then transform the function by **p**. Output the new function to **obj**.

- `obj = diff()` - differentiate the piecewise polynomial and output the resulting piecewise polynomial to `obj`.
- `obj = int()` - integrate the piecewise polynomial and output the resulting piecewise polynomial to `obj`.

LagrangeInterpolator (subclass of the **PiecewisePolynomial**)

This class creates and stores the properties of **PiecewisePolynomial** required for a Lagrange Interpolator.

Properties As well as the properties inherited from **PiecewisePolynomial**;

- `dt` - the timestep used for this function.

Constructor arguments

- `dt` - as above.
- `degree` - degree of the Lagrange interpolator.

Methods

- `varargout = padCoeffs(varargin)` - pad the coefficients of any number of **LagrangeInterpolator** instances (specified in the arguments) so that they all match the instance this function is called from.

BasisFunction (class)

This class creates and stores basis functions (or testing functions) in the form of **PiecewisePolynomial** instances which are defined on every edge of a geometry.

Properties

- `pol` - a cell matrix of size $[number\ of\ partitions \times number\ of\ edges]$ which stores the basis function(s) for each edge in the form of **PiecewisePolynomial** instances.
- `idx` - a cell array of size $[number\ of\ edges]$, each cell contains an array of indices that define which polynomials in `pol` apply to the

edge.

- **idx_table** - a matrix of size $[number\ of\ edges \times number\ of\ partitions]$. Each row of the index table represents the halfedges that the i 'th polynomial applies to (where i is the column number).

Constructor arguments

There is no constructor, instead the basis functions are created directly from the methods explained below.

Methods

- **obj = createHat(halfedges, scale)** - create hat basis functions that apply to the **halfedges** of a mesh. **scale** is a an optional boolean (set to **false** by default) which determines whether to scale the function amplitude by $1/\text{length}$, i.e. force the function to have a unit height.
- **obj = createSquare(halfedges, scale)** - create square basis functions that apply to the **halfedges** of a mesh. **scale** is a an optional boolean (set to **false** by default) which determines whether to scale the function amplitude by $1/\text{length}$, i.e. force the function to have a unit height.
- **obj = createDualSquare(dual_halfedges, scale)** - create dual square basis functions that apply to the **dual_halfedges** of a mesh. **scale** is a an optional boolean (set to **false** by default) which determines whether to scale the function amplitude by $1/\text{length}$, i.e. force the function to have a unit height.
- **obj = createDualHat(dual_halfedges, scale)** - create dual hat basis functions that apply to the **dual_halfedges** of a mesh. **scale** is a an optional boolean (set to **false** by default) which determines whether to scale the function amplitude by $1/\text{length}$, i.e. force the function to have a unit height.
- **obj = divergence(halfedges)** - outputs an instance of **BasisFunction** which has it's polynomials spatially differentiated with respect to the **halfedges** input.
- **obj = plot_basis(halfedges, basis)** - collapse the 2D geometry given by **halfedges** into a horizontal line, and plot the basis functions given by **basis** associated with each edge on the line.

computeConvolutions (function)

This function processes the temporal convolution between the Lagrange interpolator temporal-basis function (and its integrated and derivative forms) and the 2D time-domain Green's function.

Arguments

- **distances** - a matrix of distances (from source to observation).
- **intTB** - the integrated temporal basis function, as an instance of **LagrangeInterpolator**.
- **TB** - the temporal basis function, as an instance of **LagrangeInterpolator**.
- **dTB** - the derivative of the temporal basis function, as an instance of **LagrangeInterpolator**.

Outputs

- **Fh** - a matrix of convolution values of the same size as **distances** (computed with the integrated Lagrange interpolator).
- **Fs** - a matrix of convolution values the same size as **distances** (computed with the Lagrange interpolator).
- **dF** - a matrix of convolution values the same size as **distances** (computed with the derivative of the Lagrange interpolator).

RHS (class)

This class computes the right hand side i.e. the **V** vector for the 2D TDBEM which includes testing the incident field.

Properties

- **N_T** - total number of timesteps.
- **dt** - timestep.
- **geometry** - a list of halfedges in the form of **MeshBoundary.halfedges** or **MeshBoundary.dual**.
- **test_function** - the function used for testing, in the form of a **BasisFunction**.
- **display_plot** - a boolean which determines whether to plot the

wave at various points on the mesh after computation. The default is **false**.

- **excitation** - a function which evaluates the amplitude of the wave given a time and location, i.e. the **eval** an instance of an **Excitation** subclass.
- **polarization** - row vector specifying a polarization as a 2D unit vector; e.g. $[1 \ 0]$ = x-direction, $[0 \ -1]$ = negative y-direction, 1 = normal to plane (z-direction).
- **Gaussian_points** - a value to specify how many Gaussian quadrature points to use per edge. The default is 3.

Constructor arguments

- **N_T** - as above.
- **dt** - as above.

Methods

- **V = compute(tangent)** - compute the right hand side and output a matrix of size $[number\ of\ edges \times N_T]$, where **tangent** is a boolean which enables or disables taking the edge tangents into account; used for fields transverse to the plane. If **tangent** is not given, the default is **false**.

GramMatrix (class)

This class computes the Gram matrix.

Properties

- **basis_function** - the function used for sampling, in the form of a **BasisFunction**.
- **test_function** - the function used for testing, in the form of a **BasisFunction**.
- **geometry** - a list of halfedges in the form of **MeshBoundary.halfedges** or **MeshBoundary.dual**.
- **test_points** - a value to specify how many Gaussian quadrature points to use per edge. The default is 3.

Constructor arguments

There is no constructor.

Methods

- **G = compute** - compute the Gram matrix and output a matrix of size $[number\ of\ edges^2]$.

ZMatrices (class)

This class computes the 2D TDBEM matrices for **S**, **D**, **D'**, **N_h**, and **N_s**.

Properties

- **N_T** - total number of timesteps.
- **dt** - timestep.
- **c** - speed of propagation through the medium.
- **timeBasis_D** - the temporal basis function, as an instance of **LagrangeInterpolator**.
- **timeBasis_Nh** - the integral of the temporal basis function, as an instance of **LagrangeInterpolator**.
- **timeBasis_Ns** - the derivative of the temporal basis function, as an instance of **LagrangeInterpolator**.
- **basis_function_Z** - the function used for sampling fields in the z-direction, in the form of a **BasisFunction**.
- **basis_function_S** - the function used for sampling fields transverse to the plane, in the form of a **BasisFunction**.
- **test_function_Z** - the function used for testing fields in the z-direction, in the form of a **BasisFunction**.
- **test_function_S** - the function used for testing fields transverse to the plane, in the form of a **BasisFunction**.
- **outer_points_sp** - a value to specify how many Gaussian quadrature points to use per edge for the outer (testing) integral when it is at a potential singular point. Default is 50.
- **inner_points_sp** - a value to specify how many Gaussian quadrature points to use per edge for the inner (sampling) integral when it is at a potential singular point. Default is 51.
- **outer_points** - a value to specify how many Gaussian quadrature

points to use per edge for the outer (testing) integral. Default is 3.

- **inner_points** - a value to specify how many Gaussian quadrature points to use per edge for the inner (sampling) integral. Default is 4.

Constructor arguments

- **N_T** - as above.
- **dt** - as above.
- **geom_obj** - a list of halfedges in the form of `MeshBoundary.halfedges` or `MeshBoundary.dual`.
- **c** - as above.

Methods

- `[S,D,Dp,Nh,Ns] = compute(cheat)` - compute the TDBEM matrices for **S**, **D**, **D'**, **N_h**, and **N_s** and output them as **S**, **D**, **Dp**, **Nh**, and **Ns** respectively. Each matrix is of size $[number\ of\ edges^2 \times N_T]$. The **cheat** boolean can be set to true when the geometry is a cylinder with equal edge lengths. When using the **cheat**, the algorithm runs much faster because it only has to compute 1 row and column of the matrix for each timestep, then copies them to fill the rest of the matrix. If **cheat** is not defined, the default is **false**.

B

BEM C++ Implementation Manual

Because Matlab dynamically allocates memory, and doesn't support pointers or referencing, it can be slow compared to C++. Furthermore, portable code that can run in parallel on multiple threads is much more convenient using an open source C++ compiler, OpenMP, and CMake. The implementation for 2DTDBEM can be found at <https://github.com/dan-phd/2DTDBEM>.

The 2DTDBEM project repository has the following directory structure:

```
2DTDBEM ..... contains the license, readme and install script
├── 2DTDBEM ..... contains the source code and CMakeLists file
│   ├── build ..... where the built binaries go
│   ├── CMake ..... additional tools for CMake
│   ├── input ..... where the input Matlab files go
│   └── results ..... where the output Matlab files go
```

B.1 Installation

To install, use the CMakeLists file. There are dependencies on the following libraries:

- OpenMP (optional but recommended for parallel computing and faster run-times)
- Zlib (optional but recommended for compression)
- HDF5 (optional but recommended for large files)
- MatIO (required)
- Armadillo (required)

B.1.1 Linux

From a fresh install (e.g. Amazon Web Services EC2 Ubuntu), download the project to a custom directory, then run the `install.sh` script:

```
sudo apt-get install git
git clone https://github.com/dan-phd/2DTDBEM.git
cd 2DTDBEM
chmod +x install.sh
sudo ./install.sh
```

Alternatively, you can install the above libraries (in order) yourself using the standard `./configure` and `sudo make install` commands. If you don't have root privileges, you will need to install the libraries in a local folder, and use

```
./configure --prefix=/home/<local_lib_folder>
```

or

```
cmake . -DCMAKE_INSTALL_PREFIX=/home/<local_lib_folder>
```

If HDF5 is installed, MatIO should be configured using

```
./configure --with-default-file-ver=7.3
```

Once the libraries have been installed, 2DTDBEM is installed using

```
cd 2DTDBEM/build
sudo cmake ..
sudo make install
cd ..
```

The program options can then be viewed with `./bin/2DTDBEM`. If you get an error while loading the shared libraries, use

```
export LD_LIBRARY_PATH=/usr/local/lib/
```

for root users or otherwise use

```
export LD_LIBRARY_PATH=/home/<local_lib_folder>/lib
```

B.1.2 Windows

First download and unzip the `armadillo` and `MatIO` libraries to a low level directory (such as `C:\build`).

Edit the `user environment variables` for your machine to include the directories to the unzipped libraries in variables named `ARMADILLO_ROOT` and `MATIO_ROOT`.

To run the code using Visual Studio in Windows, open the `2DTDBEM.sln` file located in the top directory.

Make sure that the correct solution platform is being used (Win32 or x64) and check the following settings are configured correctly:

- *Configuration Properties* → *C/C++* → *General* → *Additional Include Directories*:
 - `$(ARMADILLO_ROOT)\include`
 - `$(MATIO_ROOT)\include`
- *Configuration Properties* → *Linker* → *General* → *Additional Library Directories*:

- \$(ARMADILLO_ROOT)\include\examples\lib_win64
- \$(MATIO_ROOT)\visual_studio\x64\Debug (or Release depending on solution configuration)
- *Configuration Properties* → *Linker* → *Input* → *Additional Dependencies*:
 - lapack_win64_MT.lib
 - blas_win64_MT.lib
 - libmatio.lib

After building the Visual Studio project but before running, make sure that `lapack_win64_MT.dll`, `blas_win64_MT.dll`, and `libmatio.dll` is copied to the runtime directory i.e. the same directory as the `2DTDBEM.exe` file.

To run the application straight from Visual Studio, you can append program arguments in *Configuration Properties* → *Command Arguments*. The program can then be run without the debugger by pressing **Ctrl + F5**.

B.2 Usage

From the 2DTDBEM directory, run the program using `./bin/2DTDBEM [options]`, where the options are:

- `-h` or `--help` - Print usage and exit.
- `-f<arg>` or `--file=<arg>` - Input mesh filename, without extension (required).
- `-t<num>` or `--timesteps=<num>` - Number of timesteps. [1000]
- `-q<num>` or `--quadrature_points=<num>` - Number of Gaussian quadrature points used on the outer integral. [25]
- `-d<num>` or `--degree=<num>` - Lagrange interpolator degree to use for the temporal convolutions. [1]
- `-s<arg>` or `--suffix=<arg>` - suffix to attach to end of result filename.
- `-c` or `--cheat` - Use cheat for faster computation (only applicable for cylinder with symmetric edge lengths).

-S or **--scattered** - Compute scattered field.

-T<args> or **--test <args>** - Perform test specified in the argument.

B.2.1 Examples

A very simple example using the input file `cyl_res21.mat`, and all defaults:

```
./bin/2DTDBEM --file cyl_res21
```

Using the input file `cyl_res21.mat`, compute the operator matrices for 300 timesteps, using 4 quadrature points, and using the cheat:

```
./bin/2DTDBEM --file=cyl_res21 --timesteps=300 --  
quadrature_points=4 --cheat
```

The same as above but reduced:

```
./bin/2DTDBEM -fcyl_res21 -t300 -q4 -c
```

Compute the scattered fields in file `scattered_mesh.mat` for 500 timesteps

```
./bin/2DTDBEM --scattered -fscattered_mesh -t500
```

The input file is a specific Matlab type file which contains boundary edges, `dt`, `c`, number of shapes, and an option to decide whether or not to use dual basis functions.

The results folder contains the output files, which have the same name as the input (plus a suffix if specified).

B.2.2 Initial test

Once all the files are copied and the libraries are installed, run the following initial test to check everything works:

```
./bin/2DTDBEM --test computeConvolutions
```

Then in the Matlab program, modify the `locationOfFolder` variable set in `BEUT.CFolder` to the directory in which the C++ program is installed; this should be one directory down from the folders `/input` and `/results`.

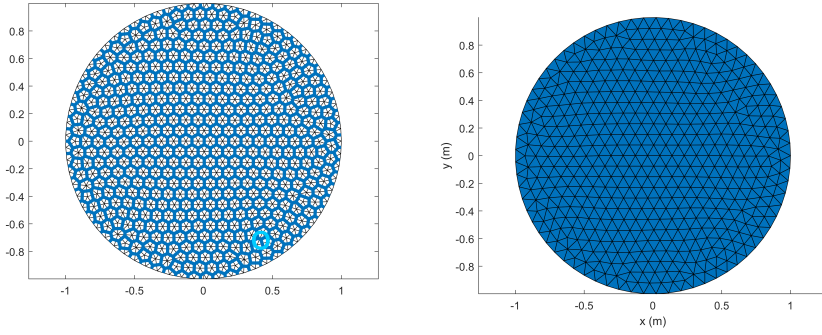
Now you can run the `BEUT.BEM.Demo.computeConvolutions` script and compare the time and accuracy for the results using Matlab and the results using C++.



BEUT Tutorial

This section will demonstrate the steps required to model a Lüneburg lens from scratch.

- 1) Download the BEUT Matlab project and the 2DTDBEM C++ project from <https://github.com/dan-phd/BEUT> and <https://github.com/dan-phd/2DTDBEM>, respectively.
- 2) Unzip both projects into a custom directory. For example, C:\tutorial\BEUT and C:\tutorial\2DTDBEM.
- 3) Open Matlab and change the path to the BEUT folder.
- 4) In Matlab, type `open BEUT.CFolder`, and change the `locationOfFolder` variable to the location of the 2DTDBEM directory.
- 5) In Matlab, type `open BEUT.Meshing.Main.CreateMesh` and make sure the script is creating a “cylinder resonator” with 1 wavelength per radius. Run the script and observe the graphical output which shows the link line mesh as shown in figure C.1a, and the output triangle mesh, which in this case should have 69 boundary edges, as shown in figure C.1b.



(a) Link line mesh, with smallest link line circled.

(b) Output triangulation.

Figure C.1: Resulting mesh using the CreateMesh Matlab script.

Observe the output in the Matlab command window which displays the ratio between the average link length and shortest link length (which from experience is most efficient below 5). The command window output also displays the location of the output files, such as:

```
Matlab file output to: C:\tutorial\BEUT\+BEUT\+Meshing\
    meshes\cyl_res69.mat
C++ file output to: C:\tutorial\2DTDBEM\2DTDBEM\input\
    cyl_res69.mat
```

- 6) To compute the BEM operators, open command prompt from the 2DTDBEM directory (C:\tutorial\2DTDBEM\2DTDBEM in this case) and type:

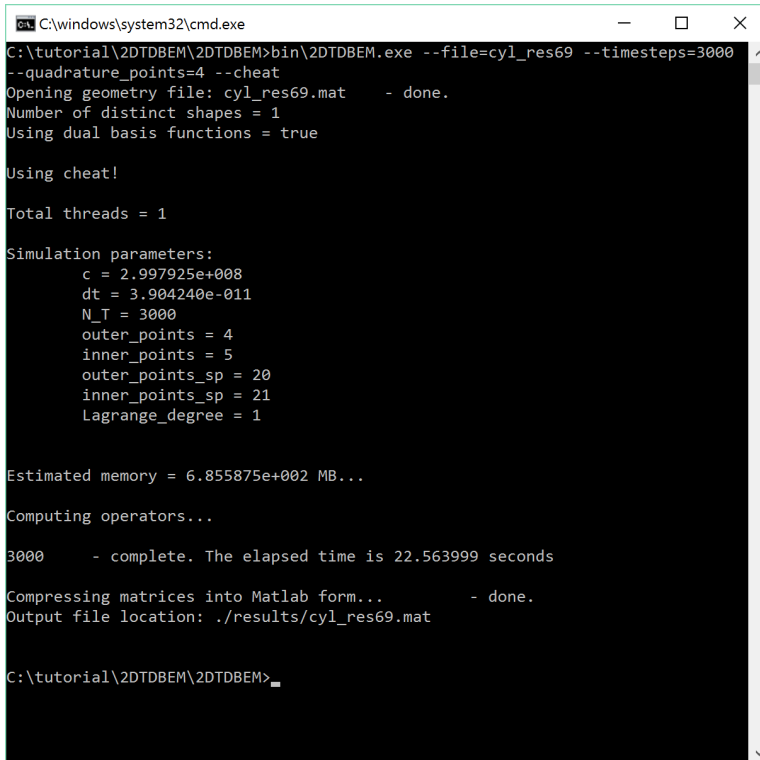
```
bin\2DTDBEM.exe --file=cyl_res69 --timestep=3000
    --quadrature_points=4 --cheat
```

Once the computation is complete, the command window should display the location of the output file (which should be in the **results** folder) as shown in figure C.2.

- 7) In Matlab, type `open BEUT.Main.ModelLuneburgLens` and follow the code through:

- (a) Load the geometry (along with `dt`, `mu0` and `eps0`):

```
filename = 'cyl_res69';
```



```
C:\windows\system32\cmd.exe
C:\tutorial\2DTDBEM\2DTDBEM>bin\2DTDBEM.exe --file=cyl_res69 --timesteps=3000
--quadrature_points=4 --cheat
Opening geometry file: cyl_res69.mat    - done.
Number of distinct shapes = 1
Using dual basis functions = true

Using cheat!

Total threads = 1

Simulation parameters:
    c = 2.997925e+008
    dt = 3.904240e-011
    N_T = 3000
    outer_points = 4
    inner_points = 5
    outer_points_sp = 20
    inner_points_sp = 21
    Lagrange_degree = 1

Estimated memory = 6.855875e+002 MB...

Computing operators...

3000    - complete. The elapsed time is 22.563999 seconds

Compressing matrices into Matlab form...    - done.
Output file location: ./results/cyl_res69.mat

C:\tutorial\2DTDBEM\2DTDBEM>
```

Figure C.2: Command prompt output after computing BEM operators.

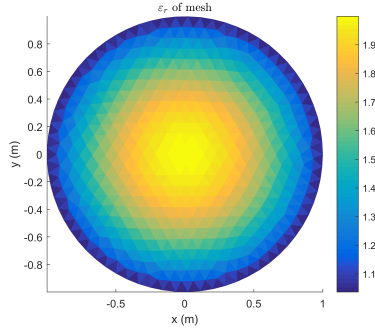


Figure C.3: Relative permittivity coverage for the Lüneburg lens.

```
global mu0 eps0;
load([fileparts(which('BEUT.Meshing.load')) filesep
    'meshes' filesep filename '.mat']);
boundary=BEUT.Meshing.MeshBoundary(mesh);
radius=max(range(vercat(boundary.halfedges.a)))/2;
c0 = 1/sqrt(mu0*eps0);
```

Make sure the `filename` variable matches the name of the mesh (which is also the same as the 2DTDBEM results file).

- (b) Define material layers, where the Lüneburg lens has an ε_r that is a function of the distance from the center:

```
CC = circumcenter(mesh.TR);
for i=1:mesh.nF
    r = norm(CC(i,:));
    eps_r(i) = (2-(r/radius).^2);
end
mu_r=1;
```

- (c) Set the materials in the `mesh` object, and plot the relative permittivity coverage:

```
mesh.setMaterial(eps_r,mu_r);
mesh.calcAdmittances;
mesh.plot_materials('eps_r')
```

The resulting figure should look similar to figure C.3.

- (d) Set the total number of timesteps and make the time vector:

```
N_T = 3000;
time = 0:dt:(N_T-1)*dt;
```


- (e) Set the excitation, in this case a Gaussian modulated sinusoidal wave:

```
min_edge_length = min(vercat(boundary.halfedges.1)
);
f_width = 0.3 * c0;
f_mod = 0.9 * c0;
direction = [1 0];
inc_wave = BEUT.Excitation.SineWave(f_width, f_mod,
c0, direction, 0.8);
V_source = inc_wave.eval(time);
```

View the excitation in the time domain (as shown in figure C.4a:

```
figure; plot(time,V_source)
title('Incident wave in the time domain'); xlabel('
time');
```

Check the stability and view the excitation in the frequency domain (as shown in figure C.4b):

```
min_edge_length = min(vercat(boundary.halfedges.1)
);
min_wavelength = c0/inc_wave.freq_response(time,
true);
if min_edge_length>min_wavelength/10
warning(['Minimum edge length (' num2str(
min_edge_length) ') should be less than a
tenth of the minimum wavelength ('num2str(
min_wavelength) ')'])
end
```

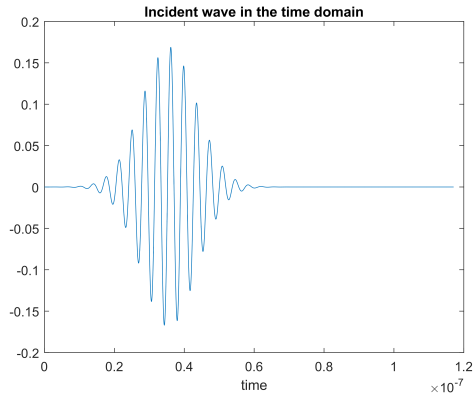
A warning may appear which can be safely ignored if the minimum edge length is not less than, but close to, a tenth of the minimum wavelength.

- (f) Set the probe positions, which will be located on the observed halfedges:

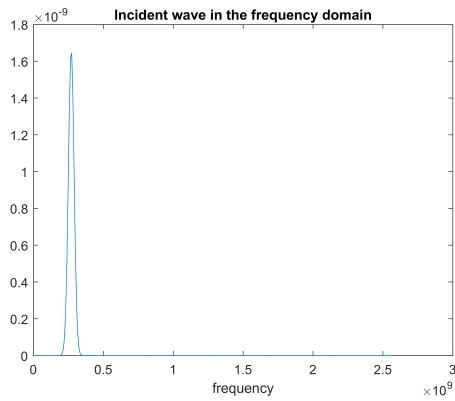
```
observation_edges = [117 1048];
mesh.plot_halfedge(observation_edges);
```

In this case, the observed points are located on the boundary at the far left and far right sides of the cylinder as shown in figure C.5.

- (g) Compute the hybrid operators (using the C++ operator file) and perform the main timestepping algorithm:



(a) In the time domain.



(b) In the frequency domain.

Figure C.4: Incident wave.

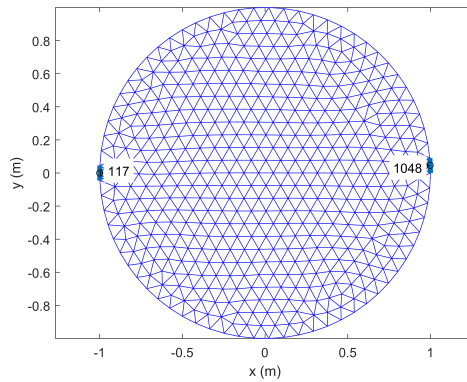


Figure C.5: Locations of the observation points.

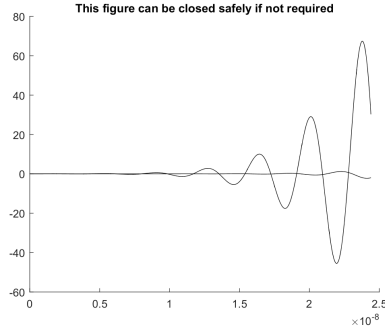


Figure C.6: Real-time simulation plot.

```
source_edge = observation_edges(1);
operator_file = matfile([BEUT.CFolder filesep '
results' filesep filename '.mat']);
[mesh, M_TM, J_TM] = BEUT.Main.MOT(mesh, boundary,
operator_file, observation_edges, time, mu0, 0,
0, V_source, source_edge);
```

The 7th and 8th arguments represent the electric and magnetic incident plane waves, which are set to zero in this case because there is no plane wave present in this simulation. The real-time animation figure, similar to the one shown in figure C.6, can be closed if not required (simulations run slightly faster without plotting at every timestep).

- (h) Once the simulation has completed, you can plot the field at the observation points specified previously using:

```
tstop = size(mesh.fields.E_z,2);
figure; plot(time(1:tstop),mesh.fields.E_z(
observation_edges,1:tstop))
entries = cell(1,numel(observation_edges));
for i=1:numel(observation_edges)
    entries(i) = {sprintf('E_z at halfedge %i',
observation_edges(i))};
end
legend('String',entries);
```

The resulting figure should look similar to the one shown in figure C.7.

- (i) For an animation of the electric field inside the object (the UTLM region), use:

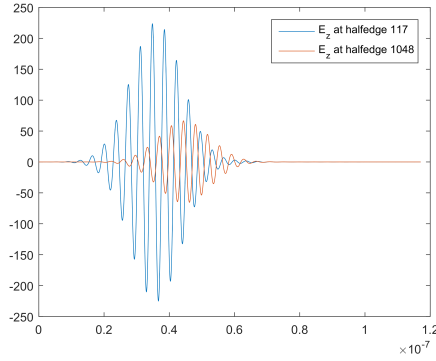


Figure C.7: Locations of the observation points.

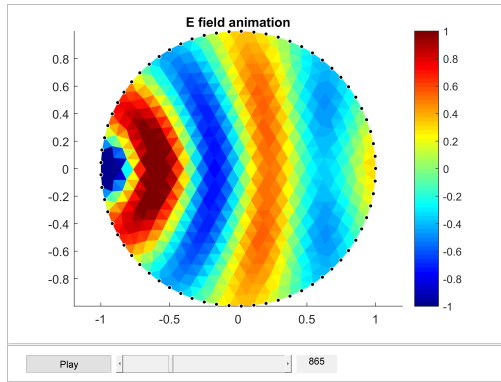


Figure C.8: Animation of the internal fields paused at timestep 865.

```
mesh.animate('E')
```

The animated figure that appears allows you to play, pause and skip frames, as shown in figure C.8.

- (j) For scattered fields outside of the UTLM region, observation points must be defined so that the BEM operators can act upon those points. We can output a file which will contain this information using a structured set of points. BEM will also need to know the surface current densities and whether or not to use dual basis functions:

```
[X,Y] = meshgrid([-1.5:0.2:2],[-1.5:0.2:1.5]);
x_coords = X(:); y_coords = Y(:);
M = mesh.fields.E_z(mesh.mesh_boundary,:);
```

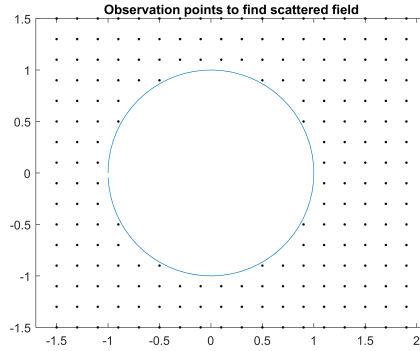


Figure C.9: Locations of the observation points to be used when finding the scattered field outside of the UTLM region.

```
J = -mesh.fields.H_xy(mesh.mesh_boundary,:);
dual = true;
c_file = [BEUT.CFolder filesep 'input' filesep
          filename '_scattered.mat'];
in_scatterer = BEUT.BEM.Main.
saveScatteredFieldPoints(mesh,x_coords,y_coords
,M,J,dual,c_file);
```

A figure should then be output which shows the object outline and observation points around it, similar to figure C.9. The command window output indicates the number of points that will be computed; the less points, the less memory required and the quicker the computation. The output also reveals the filename that is to be input into the 2DTDBEM program, for example:

```
Number of grid points: 208
C++ file output to: C:\tutorial\2DTDBEM\2DTDBEM\
input\cyl_res69_scattered.mat
```

- 8) To compute the scattered fields outside of the object, open command prompt from the 2DTDBEM directory and type:

```
bin\2DTDBEM.exe --file=cyl_res69_scattered --timestep
=3000 --scattered
```

- 9) Return to the Matlab script `BEUT.Main.ModelLuneburgLens`, and use the following code to animate the BEM scattered fields:

```
operator_file = matfile([BEUT.CFolder filesep 'results'
```

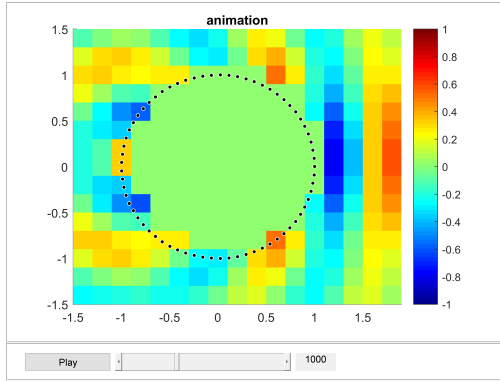


Figure C.10: Animation of the external fields paused at timestep 1000.

```

        filesep filename '_scattered.mat']]);
E_s = BEUT.BEM.Main.organizeScatteredField(
    operator_file, X, in_scatterer );
BEUT.animate_fields(2,'domain',X,Y, 'animation',E_s/max
    (max(max(E_s))), 'overlay',vertcat(boundary.
    halfedges.a),'dimensions',2, 'skipTimesteps',10, '
    max_amplitude',1,'min_amplitude',-1);

```

The animation window will look similar to figure C.10.

- 10) To plot the fields inside and outside the scatterer at a particular timestep (in this case we will specify the timestep as 1000), use the following:

```

BEUT.Main.plotFields( filename,mesh,X,Y,in_scatterer
    ,1000,true )

```

The resulting output will look similar to figure C.11, where the external scattered field is now interpolated across the structured mesh for a more visually appealing image.

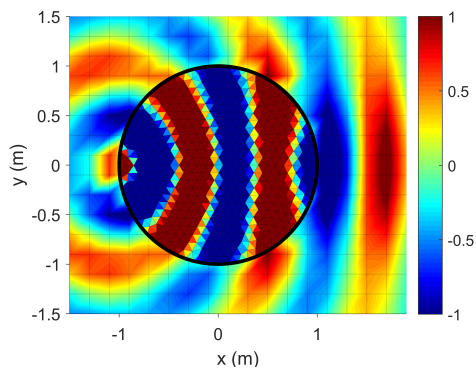


Figure C.11: The electric field plot inside and outside of the scatterer at timestep 1000.